ICIMP 2016

The Eleventh International Conference on Internet Monitoring and Protection

ISBN: 978-1-61208-475-6

May 22 - 26, 2016

Valencia, Spain

ICIMP 2016 Editors

Jaime Lloret Mauri, Polytechnic University of Valencia, Spain
Mario Freire, University of Beira Interior, Portugal

ICIMP 2016

Foreword

The Eleventh International Conference on Internet Monitoring and Protection (ICIMP 2016), held between May 22-26, 2016, in Valencia, Spain, continued a series of special events targeting security, performance, vulnerabilities in Internet, as well as disaster prevention and recovery.

The design, implementation and deployment of large distributed systems are subject to conflicting or missing requirements leading to visible and/or hidden vulnerabilities. Vulnerability specification patterns and vulnerability assessment tools are used for discovering, predicting and/or bypassing known vulnerabilities.

Vulnerability self-assessment software tools have been developed to capture and report critical vulnerabilities. Some of vulnerabilities are fixed via patches, other are simply reported, while others are self-fixed by the system itself. Despite the advances in the last years, protocol vulnerabilities, domain-specific vulnerabilities and detection of critical vulnerabilities rely on the art and experience of the operators; sometimes this is fruit of hazard discovery and difficult to be reproduced and repaired.

System diagnosis represent a series of pre-deployment or post-deployment activities to identify feature interactions, service interactions, behavior that is not captured by the specifications, or abnormal behavior with respect to system specification. As systems grow in complexity, the need for reliable testing and diagnosis grows accordingly. The design of complex systems has been facilitated by CAD/CAE tools. Unfortunately, test engineering tools have not kept pace with design tools, and test engineers are having difficulty developing reliable procedures to satisfy the test requirements of modern systems. Therefore, rather than maintaining a single candidate system diagnosis, or a small set of possible diagnoses, anticipative and proactive mechanisms have been developed and experimented. In dealing with system diagnosis data overload is a generic and tremendously difficult problem that has only grown. Cognitive system diagnosis methods have been proposed to cope with volume and complexity.

Attacks against private and public networks have had a significant spreading in the last years. With simple or sophisticated behavior, the attacks tend to damage user confidence, cause huge privacy violations and enormous economic losses.

The CYBER-FRAUD track focuses on specific aspects related to attacks and counterattacks, public information, privacy and safety on cyber-attacks information. It also targets secure mechanisms to record, retrieve, share, interpret, prevent and post-analyze of cyber-crime attacks.

Current practice for engineering carrier grade IP networks suggests n-redundancy schema. From the operational perspective, complications are involved with multiple n-box PoP. It is not guaranteed that this n-redundancy provides the desired 99.999% uptime. Two complementary solutions promote (i) high availability, which enables network-wide protection by providing fast recovery from faults that may occur in any part of the network, and (ii) non-stop routing. Theory on robustness stays behind the attempts for improving system reliability with regard to emergency services and containing the damage through disaster prevention, diagnosis and recovery.

We take here the opportunity to warmly thank all the members of the ICIMP 2016 Technical Program Committee, as well as all of the reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to ICIMP 2016. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the ICIMP 2016 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that ICIMP 2016 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the field of Internet monitoring and protection.

We are convinced that the participants found the event useful and communications very open. We hope that Valencia provided a pleasant environment during the conference and everyone saved some time to enjoy the charm of the city.

ICIMP 2016 Chairs:

ICIMP General Chair
Jaime Lloret Mauri, Polytechnic University of Valencia, Spain

ICIMP Advisory Committee
Go Hasegawa, Osaka University, Japan
Sathiamoorthy Manoharan, University of Auckland, New Zealand
Jaime Lloret Mauri, Polytechnic University of Valencia, Spain
Constantion Paleologu, University ‘Politehnica’ Bucharest, Romania
Michael Grottke, University of Erlangen-Nuremberg, Germany
William Dougherty, Secern Consulting - Charlotte, USA

ICIMP Industry/Research Chairs
Mohamed Eltoweissy, Virginia Military Institute and Virginia Tech, USA
Nicolas Fischbach, COLT Telecom, Germany
Emir Halepovic, AT&T Labs - Research, USA
Miroslav Velev, Aries Design Automation, USA
Steffen Wendzel, Fraunhofer FKIE, Germany
Artsiom Yautsiukhin, National Council of Research, Italy

ICIMP 2016

COMMITTEE

ICIMP General Chair
Jaime Lloret Mauri, Polytechnic University of Valencia, Spain

ICIMP Advisory Committee

Go Hasegawa, Osaka University, Japan
Sathiamoorthy Manoharan, University of Auckland, New Zealand
Jaime Lloret Mauri, Polytechnic University of Valencia, Spain
Constantin Paleologu, University ‘Politehnica’ Bucharest, Romania
Michael Grottke, University of Erlangen-Nuremberg, Germany
William Dougherty, Secern Consulting - Charlotte, USA

ICIMP Industry/Research Chairs

Mohamed Eltoweissy, Virginia Military Institute and Virginia Tech, USA
Nicolas Fischbach, COLT Telecom, Germany
Emir Halepovic, AT&T Labs - Research, USA
Miroslav Velev, Aries Design Automation, USA
Steffen Wendzel, Fraunhofer FKIE, Germany
Artsiom Yautsiukhin, National Council of Research, Italy

ICIMP 2016 Technical Program Committee

Jemal Abawajy, Deakin University - Victoria, Australia
Rifaat Abdalla, King Abdulaziz University, Saudi Arabia
Mohd Taufik Abdullah, Universiti Putra Malaysia, Malaysia
Jihad Mohamad Al Ja'am, Qatar University - College of Engineering, Qatar
Javier Barria, Imperial College London, UK
Olga Battaïa, ISAE-Supaero, Toulouse, France
Fadila Bentayeb, University of Lyon, France
Lasse Berntzen, University College of Southeast, Norway
Jonathan Blackledge, Dublin Institute of Technology, Ireland
Matthias R. Brust, University of Central Florida, USA
Christian Callegari, University of Pisa, Italy
Eduardo Cerqueira, Federal university of Para, Brazil
Hugo Coll, Universidad Politécnica de Valencia, Spain
Christopher Costanzo, U.S. Department of Commerce, USA
Jianguo Ding, University of Skövde, Sweden
Dimitris Dranidis, CITY College - International Faculty of the University of Sheffield, Greece
Mohamed Eltoweissy, Virginia Military Institute and Virginia Tech, USA
Nicolas Fischbach, COLT Telecom, Germany
Ulrich Flegel, SAP Research - Karlsruhe, Germany
Alex Galis, University College London, UK

Table of Contents

Code-Stop: Code-Reuse Prevention By Context-Aware Traffic Proxying

Terrence OConnor and William Enck
Department of Computer Science
North Carolina State University
Raleigh, North Carolina, USA
Email: tjoconno@ncsu.edu, enck@cs.ncsu.edu

***Abstract*—This paper introduces a network and host-based cooperative system for defending against code-reuse attacks that bypass exploit mitigation strategies. While the combination of address space layout randomization (ASLR) and data execution prevention (DEP) provide the means for mitigating exploitation, attackers routinely bypass these mechanisms by borrowing code from shared libraries that lack the same protections or by abusing memory leaks. This paper illustrates the ability to identify code-reuse attacks through cooperation between the traffic proxy and destination host. With the context of the host, the network has the ability to prevent code-reuse, and ultimately, exploitation. Through experimentation, we demonstrate that our cooperative system can effectively defeat a wide variety of code-reuse attacks, including newer attack vectors such as Just-in-Time-Flash or jump-oriented gadgets. Our experiments indicate our prototype is compatible with popular software such as Internet Explorer, Adobe Reader, and Microsoft Office applications and proved successful mitigating code-reuse attacks.**

Keywords–code-reuse attacks; return-oriented programming; intrusion prevention system; proxy.

I. INTRODUCTION

Identifying and defending against exploits in the wild is an ongoing challenge. While system protections such as Address Space Layout Randomization (ASLR) and Data Execution Prevention (DEP) make vulnerabilities more difficult to exploit, they can be bypassed using code-reuse attacks (e.g., Return Oriented Programming [ROP], Jump Oriented Programming [JOP], SigReturn Oriented Programming [S-ROP], and Just In Time Return Oriented Programming [JIT-ROP]).

In 2012, Microsoft's BlueHat Challenge [1] awarded over a quarter of a million dollars to three solutions that defended against ROP. Within a year, Shacham et al. [2] successfully bypassed every ROP protection that had been awarded a prize. Additionally, after Microsoft integrated the BlueHat Challenge winner solutions into their commercial product The Enhanced Mitigation Experience Toolkit (EMET), DeMott [3] demonstrated separate methods for bypassing all twelve system protections included in EMET. Most defense mechanisms have focused exclusively on the host by either compiling gadget-free binaries, protecting critical functions [4], or performing runtime randomization [5], control flow analysis [6], or system caller checking [7] [8] [9]. However, all have shown weaknesses by making the potential victim responsible for managing its own safeguards.

In contrast to these prior approaches, we propose a system for defending against code-reuse attacks that pushes the defense to the network but still relies on the host to provide the active context of the attack. We focus exclusively on preventing the code-reuse attacks (e.g., ROP, JOP, S-ROP, JIT-ROP) used to bypass system protections (e.g., ASLR, DEP). The key idea behind our solution is the concept of *parameter suspicion*. We make the assumption that a code-reuse attack will be used to bypass memory protections by allocating a protected region of memory as executable. Further, we make the assumption that a code-reuse attack will borrow code in order to execute a memory-related function. Instead of monitoring the specific function calls that may allocate or change memory protections, we attempt to identify the parameters used by the function call. Parameter suspicion identifies the generic behavior that occurs prior to a code-reuse attack. It identifies when the attacker abuses memory to load data into registers, functioning as parameters for a call to a critical system call or function.

In this paper, we propose and implement Code-Stop, a collaborative system that protects an application from code-reuse attacks that bypass system protections. Code-Stop relies on parameter suspicion to identify a code-reuse attack. We implement Code-Stop on top of an existing traffic proxy in communication with destination hosts. We observe that Code-Stop can prevent modern client-side attacks at the network layer only with the emulated context being provided by the host. We demonstrate that prevention can occur with minimal overhead by reducing the critical area of traffic that must be tested. Our prototype scans for potential code-reuse attacks in PDF documents, JavaScript, Adobe Flash, and Microsoft Office documents, which account for 72.9% of vulnerable file types [10].

This paper makes the following contributions:

- We design and implement Code-Stop to protect against the broader threat of client-side attacks that use code-reuse attacks. Code-Stop allows the traffic proxy to make context-aware decisions about malicious traffic based on the emulated impact on the destination host.
- We propose the technique of *parameter suspicion* to identify code-reuse attacks with low false positives. Parameter suspicion emulates potential gadgets to determine the impact on the general purpose registers used as parameters when calling Windows API functions. Specifically, parameter suspicion identifies the parameters used for a Windows API function that allocates or changes memory as executable.
- We evaluate the accuracy, performance overhead, scalability and coverage of Code-Stop. We observe Code-Stop's ability to prevent code-reuse attacks without producing false positives with a large set of known malware-free files. Code-Stop's performance overhead and delay scales with typical proxy configurations and anti-virus scanning proxy solutions. We evaluate that Code-Stop can detect a wide range of code-reuse attacks without modification.

The remainder of this paper is as follows. Section II provides a background on memory protection mechanisms and code-reuse attacks. Section III examines the challenges with preventing code-reuse attacks. Section IV provides an overview of our solution. Section V examines the design of our prototype solution. Section VI evaluates our prototype, Code-Stop. Section VII discusses the limitations and future work. Section VIII discusses recent related work in the field of preventing code-reuse attacks, and Section IX concludes.

II. Background And Motivation

To understand the defense system presented in this paper, we must review common mitigation strategies, including ASLR, DEP, and compile-time, run-time, and network-layer exploit prevention mechanisms.

Address Space Layout Randomization: Implemented in early 2002, ASLR provided one of the earliest means to decrease the effectiveness of an exploit. ASLR prevents an attacker from using a predictable and pre-calculated virtual address for code reuse in an exploit. ASLR can randomize the location of code by randomizing the starting address of dynamic-linked libraries, the base address of the heap, or the location of routines and static data in the executable [11] [12] [13]. Initial research targeting ASLR implementations studied the effectiveness of defeating the entropy of code randomization [2]. However, attackers found it far more useful to bypass ASLR entirely. Notably, the initial release of Windows Vista Service Pack 0 only randomized the base address of executables and dynamic link libraries [14]. The poorly implemented Vista design effectively allowed attackers to bypass ASLR by partially overwriting the address offset without overwriting the randomized base address. Another common means for bypassing ASLR borrows code from dynamic link libraries (DLL) that lack ASLR. Several recent attacks in the wild have relied upon using DLLs without ASLR [15] [16]. Attackers routinely execute these attacks by forcing the application to load a DLL that implements extra functionality.

Data Execution Prevention: The 2004 release of Windows XP Service Pack 2 introduced the DEP security feature [17] [18]. In Windows OS, hardware DEP works similar to Linux W$\oplus$X, which uses the non-executable (NX) bit to mark memory as executable. Under W$\oplus$X or DEP, memory may be executable or writable, but not both [19]. This isolation of memory mitigates control flow hijacking by preventing stack-based buffer overflows. Combined with ASLR, DEP defeated control flow hijacking on the Windows OS until Shacham [20] and Litchfield [17] proposed the first code-reuse attacks.

Code-Reuse Attacks: In 2005, Litchfield proposed the first means of defeating DEP by returning to the VirtualAlloc() function [17]. Litchfield borrowed heavily from a Linux technique known as to return to libc, which replaced the return address on the stack with the address from a function call borrowed from the libc library. Return-Oriented-Programming (ROP) expands upon return to libc by chaining a series of borrowed code snippets together to execute a specific purpose. Under Windows, ROP often overwrites the return address with a chain of addresses that point to borrowed code inside shared libraries. ROP chains these borrowed fragments of code together to disable the DEP security mechanisms [21]. While the application's sandbox may implement memory protections, attackers often dynamically load shared libraries in order to borrow code and escape the sandbox of protection.

Under ROP, each small fragment (gadget) borrows a small piece of code that is followed by a return. The variable x86 instruction length eases the difficulty of gadget discovery, since gadgets can be borrowed from the offset of a logical address. The assembly of gadgets typically disables protection mechanisms in order to allow malicious shell-code to execute. This commonly involves placing specific values into general purpose registers before calling a Windows API function that disables the security protection of DEP.

Our solution to preventing code-reuse relies on the fact that an attacker must use gadgets to load these values into registers before calling the API function. In this way, we can observe the generic behavior in order to identify attacks. The next section examines the challenges in preventing against code-reuse attacks.

Host and Network Based Cooperative Defense: In Section VIII, we discuss the shortcomings of previous host-only or network-only defense mechanisms. We argue that the shortcomings of the host-layer and network-layer defenses can be addressed by using the network to defend with the context of the host. Previous work has examined emulating arbitrary data at the network to determine if it is part of a payload of an attack. However, code-reuse attacks rely on borrowing code from specific memory addresses. Since these addresses vary between application and operating system versions, the network must be aware of the dynamic context of the host to successfully identify a code-reuse attack.

III. Challenges

Our prototype, Code-Stop, hardens the security of the host under protection and provides new opportunities to identify attacks in progress. However, enabling this protection introduces challenges that we address:

C-1 *Ability to identify dynamic sandbox escapes.* One method commonly used by attackers forces an application to load a library lacking the same protection mechanisms as the application, so that the attacker escapes the sandbox of protection. This can be seen in recent attacks against Internet Explorer (with the `hxds.dll` bypass) and Adobe Reader (with `icuncnv36.dll` bypass.) [22], [23] Content that forces dynamic loading presents an interesting challenge for a host-based context emulator. In Section V-C, we introduce the concept of *disarmed reading*, which removes the content for a code-reuse attack and allows the application to determine whether the suspected file forces the loading of a shared library without ASLR. Preventing dynamic attacks is where our approach notably differs from previous approaches [24].

C-2 *Ability to seamlessly protect different application and operation system versions and configurations.* The address space layout protection mechanisms and fixed address space used in various bypass mechanisms differ within versions and configurations. Therefore, the proxy must dynamically construct the context of the attack to detect attacks and not be overwhelmed with a range of false positives. In Section V-B, we outline the design of the *Parameter suspicion* technique that uses the context gained

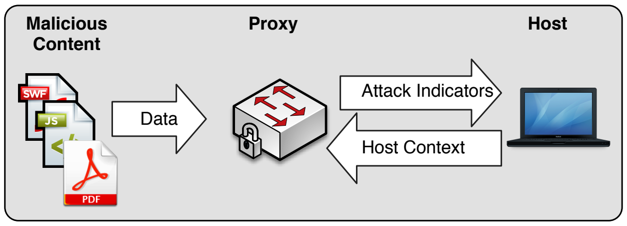


Figure 1. Overview of Code Stop

from the host to make an informed decision about the malicious nature of content.

C-3 *Impact on application performance.* Security and performance must always be closely balanced. Code-Stop protects client-side applications (Adobe Reader, Internet Explorer, Microsoft Office) from attacks where the content is rendered over the network. In Section V-A, we introduce the concept of parsing and scanning only the most absolutely necessary critical-space in affected file-formats (e.g., HTML, SWF, PDF, and DOCX).

IV. Approach

Our goal is to prevent the exploitation of a client side application by identifying the presence of code-reuse attacks in arbitrary data, such as an HTML document, PDF document stream, or Flash object. We focus on these document types since they account for 72.9% of malicious documents used in exploit kits in 2014 [10]. To successfully identify an attack, each input must be inspected with regard to the actual file format and structure to reconstruct how that data would be allocated into application memory. Figure 1 depicts the high level overview of our Code-Stop prototype. Our prototype is implemented on a network proxy, that scans suspected files to identify valid addresses that correspond to code-reuse attacks. Our approach differs from previous approaches because it includes dynamic context from the host under attack to determine if suspected data is part of a code-reuse attack.

Our hybrid approach combines the benefits of both the network layer and host layer to effectively identify and mitigate code-reuse attacks. We discuss the benefits of several host-based defenses [7] [8] [25] [26] and network-based defenses [24] [27] [28] [29] in Section VIII. However, few works have examined the benefits of combining the network and host together in defense of code-reuse attacks. Tzermias et al. [30] presented a method for the identification of ROP payloads in arbitrary data such as network traffic. However, their work failed to address the dynamic context of the host. The dynamic context of the host proves extremely important to monitor as an application can be forced to load shared libraries by processing an arbitrary document. Our work improves upon the design of Tzermias et al. by adding dynamic context of the host. Several optimizations arise out of our shared approach from Tzermias et al. By adding the network layer into the host defense, we store the record of known malicious documents. Further, caching the result of known-malicious documents allows the network layer to extend protection to hosts without our prototype software.

Assumptions and Threat Model:

We make two general assumptions in our prototype design:

1) We implement our Code-Stop prototype on the 32-bit architecture instruction set. We make the assumption that expanding our prototype to support a 64-bit architecture will only decrease the probability of false alarms. Section VI-B discusses the probability of false alarms and further explains this assumption. Further, the vast availability of exploits and ROP Chains for 32-bit applications provided for better testing of our prototype.
2) We do not address de-obfuscation as a topic for this paper. Rather, our prototype relies upon pdf-parser [31] and jsunpack [32] as means for de-obfuscating content. We make the general assumption that the proxy can de-obfuscate content or simply block heavily obfuscated content as already malicious in nature. Previous works have addressed de-obfuscating malicious code from PDF documents [33] and browser downloads [34]. Further, Section V-A discusses Code Stop's design for parsing content, supporting this assumption. Further, we expand on the limitation of de-obfuscation in Section VII. Future work may examine de-obfuscation of malicious content and the likelihood obfuscated content is benign.

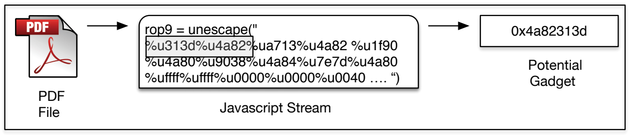
We make the following assumptions in our threat model. The adversary can exploit (i.e., control the flow of execution) of a client application (under our protection). Further, the adversary has the ability to read or infer randomized memory from those binary and shared libraries. However, we assume the attacker must bypass both DEP and ASLR to complete their exploit and execute a payload (e.g., download a remote access toolkit, add a user, or disable processes.) The trusted computing base (TCB) includes the network proxy software that parses the potential gadgets and the host application that determines the context of the gadgets. We trust that the context determined by the host application has not been altered by a malicious administrator. Finally, we assume that a trusted network channel exists between the host and the network proxy. The channel is available and preserves the traffic integrity between the proxy and the host. Given the above-described challenges, assumptions and threat model, the next section examines the design in detail.

V. Design

The following section describes the design of our prototype, Code-Stop. In Section V-A, we address how the network proxy parses the critical space of files to identify gadgets used in Code-Reuse attacks. Section V-B proposes the concept of *parameter suspicion* to identify code-reuse attacks in progress by using the emulated context of the host. Section V-C details *disarmed reading*, which disarms a malicious file in order to safely identify mitigation bypass techniques. Section V-D provides an example to illustrate how our prototype prevents an attack. Finally, Section V-E discusses how to extend Code-Stop to identify other generic exploit behaviors and operating systems.

A. Parsing Potential Gadgets

In our prototype design, the network proxy runs an application that parses arbitrary data for indicators of a potential code-reuse attack. By de-obfuscating content and further de-compiling and de-constructing specific file formats, the prototype looks for arbitrary data that represents 32-bit memory



```
rop9 = unescape("
%u313d%u4a82%ua713%u4a82 %u1f90
%u4a80%u9038%u4a84%u7e7d%u4a80
%uffff%uffff%u0000%u0000%u0040 .... “)
```

Figure 2. PDF streams containing ROP gadgets

TABLE I. REGISTER VALUES FOR COMMON ROP CHAINS

Register	VirtualAlloc()	SetInfoProcess()	VirtualProtect()
EAX	Ptr to VirtualAlloc	SizeOf 0x00000004	Ptr to VirtualProtect
EBX	dwSize 0x00000001	NtCurrentProcess 0xffffffff	dwSize 0x00000001
ECX	flProtect 0x00000040	&ExecuteFlags Ptr to 0x00000002	Writable Address
EDX	flAllocationType 0x00001000	ProcessExecuteFlags 0x00000022)	NewProtect 0x00000040

addresses or references to a variable that would contain a 32-bit address. A 32-bit memory address can be constructed many ways in memory and our parser matches regular expressions for several methods for allocating arbitrary data as 32-bits.

Consider Figure 2 for how our prototype parses a potential gadget out of a PDF File. In the example, our network proxy parser application uses pdf-parser [31] to read the embedded JavaScript inside the document. The parser then matches a regular expression for a unicode string containing two characters. Since unicode strings are 16-bits wide, two unicode characters are commonly used by attacks to represent a 32-bit memory address. The parser matches the potential gadget 0x4a82313d against the unicode string %u313d %u4a82 and passes the potential gadget to the paramater suspicion classifier.

For our prototype, we implemented the parser to examine browser traffic, including Internet Explorer Javascript, Adobe Flash Vector Objects (Actionscript) and PDF Files containing Javascript. While there are several other client side applications - we implemeted our prototype to cover the client applications that presented the largest surface area for recent exploits seen in the wild. In 2015, Trend Micro observed that Internet Explorer, Flash, and Adobe PDFs vulnerabilities accounted for 72.9% of the exploits used in the top nine exploit kits [10].

Within those specific file formats, the parser matches JavaScript's binary strings (BSTR) and Adobe Flash Objects containing ActionScript code. As illustrated with the PDF JavaScript, an attacker can cleanly construct a gadget for Internet Explorer JavaScript with a binary string of two unicode characters that refer to a 32-bit address. Although Internet Explorer 9 removed BSTR functionality, Yu [35] discovered a method for calling the earlier version of JavaScript on newer browsers by adding an HTML compatibility tag. Our prototype also decompiles Adobe Flash files and matches the embedded ActionScript code for gadgets. ActionScript code offers similar means to allocate arbitrary data as 32-bit addresses [36]. With this understanding of how our Code-Stop prototype parses allocated gadgets, the next section examines how Code-Stop detects the generic behavior of code-reuse attacks.

B. Parameter Suspicion Classifier

In order for a code-reuse attack to allocate or change memory protections, it must make a call to a Windows API function that manages memory (e.g., VirtualAlloc, VirtualProtect, SetInfoProcess.) Calling functions that allocate or change memory require parameters such as the location of memory, size, and bitwise value for new memory protections. These parameters must be placed into data registers in order to be pushed onto the call stack as parameters to the function. Rather than targeting specific API function calls, we propose the idea of a *parameter suspicion* classifier. The parameter suspicion classifier runs entirely on the host to emulate potential gadgets. Parameter suspicion determines if the instructions from the suspected gadgets load values into multiple general purpose registers. Parameter suspicion overcomes the limitations of most code-reuse defenses by identifying the anomalous characteristics of a bypass, rather than looking for specific dangerous function or system calls. We expand upon how and why it works, and the probability for false positives.

To understand how parameter suspicion works, consider Figure 3. The ROP gadgets placed onto the stack refer to instructions in the `msvcr71.dll` used by Java 1.6. The gadgets are part of a larger chain used to execute the VirtualProtect() function. After the proxy has parsed and sent potential gadgets to the host, the host emulates the effects and determines the chain loaded values into the `ECX`, `EBX`, and `EDX` registers. The key insight of parameter suspicion is that we do not need to trap the exact jump or call to the VirtualProtect() function. Rather, we can detect the gadgets placing parameter values into ECX, EBX, and EDX, respectively, to match the lpAddress, dwSize, and flNewProtect parameters required by VirtualProtect(). Using our technique we can identify any generic function call used to bypass DEP/ASLR.

We expand this understanding to other critical functions used to bypass DEP. Table I depicts the register values allocated for the VirtualAlloc(), SetInfoProcess(), and VirtualProtect() functions from ROP Chains generated by the Mona.py toolkit (a commonly used tool to automatically build ROP Chains) [37]. Note that each function requires a minimum of three parameters in addition to a pointer to the critical function. In fact, most of the 50 critical functions checked by EMET require a minimum of three parameters. Functions that allocate new memory and mark it as executable (e.g., VirtualAlloc, VirtualProtect) commonly use parameters of the following form: (1) a bitwise value for new memory protections, (2) an address to which one might write shellcode, and (3) a size of the memory allocated to the new region. Ultimately, all functions that must accomplish anything of value require multiple parameters.

Parameter suspicion does not produce significant false positives, because the probability that a data value matches a gadget address is very small. Consider the protection of Adobe Reader as an example. Assume that `icucnv36.dll` is the only current library available to bypass ASLR and DEP for the Adobe Reader application. The probability that an arbitrary eight character string corresponds to a single `POP EAX; RET` sequence of instructions is represented in Fig. 1.

$$P(R_{EAX}) = \left(\frac{22}{94}\right)^8 \cup \left(\frac{294,912}{2^{32}}\right) \cup \left(\frac{28}{294,912}\right) \quad (1)$$

We calculate this probability given that only 94 print-

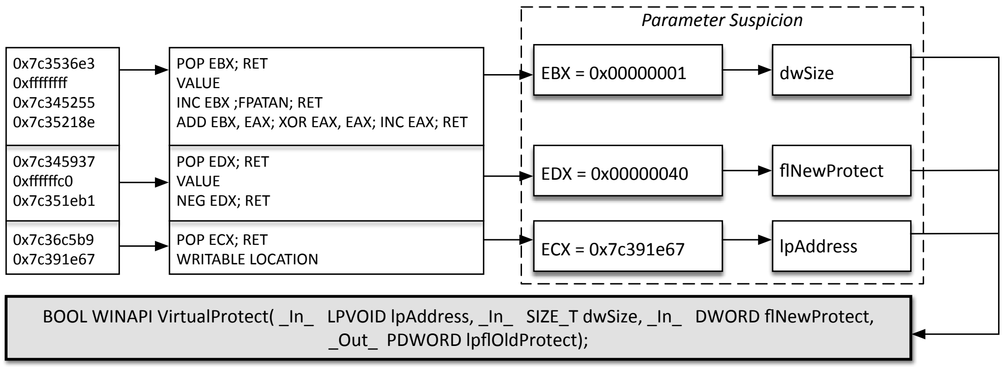


Figure 3. Parameter Suspicion used to identify msvcr71.dll ROP chain

TABLE II. LOADCONSTANT GADGETS FROM COMMON DLLS

DLL	Application	Size (Bytes)	Pop EAX	Pop EBX	Pop ECX	Pop EDX
icucnv36.dll	Adobe Reader	294,912	28	455	237	1
vgx.dll	Adobe Flash	732,672	55	699	146	3
msvcrt.dll	Visual C++ Runtime	184,320	86	358	164	22
msvcr71.dll	Java	233,472	46	234	258	21
hxds.dll	MS Office	564,224	33	481	345	7
PEhelper.dll	IBM Forms	103,936	6	78	74	0

able ASCII characters exist and only 22 of them (0-9,A-F,a-f) correspond to a memory address space. Further, the `Icucnv36.dll` only has 294,912 unique memory locations that point to executable code. And finally, only 28 of those unique locations point to a `POP EAX; RET` gadget. Our parameter suspicion classifier only matches when a chain of gadgets loads constants into three or more separate memory registers. While other operands exist (`XCHG, MOV`), the probability remains extremely small that three of these instructions will be randomly constructed from an arbitrary data stream.

To understand how this applies to other applications, examine the results in Table II. These results show the frequency of `POP REG; RET` instructions in some common shared libraries discovered by the the Metasploit *msfrop* tool (a common tool used by hackers to find instructions for code-reuse attacks.) Next, we examine how *disarmed reading* augments our classifier to determine when exploits attempt to bypass standard mitigations by loading shared libraries at runtime.

C. Disarmed Reading

One method commonly used by attackers forces an application to load a library lacking the same protection mechanisms as the application, so that the attacker escapes the sandbox of protection. Content that forces dynamic loading presents an interesting challenge for a host-based context emulator. The emulator must be aware of the addresses of all shared libraries that can be loaded dynamically by arbitrary data. *Disarmed reading* allows parameter suspicion to determine when an arbitrary file forces a protected application to load a shared library without the same protection mechanisms (e.g., a library without ASLR). To achieve this effect, disarmed reading removes suspected gadgets from the formatted file and allows the host to render the file hidden to the user. This removal includes PDF streams, JavaScript memory allocations of BSTRs, and ActionScript dynamic content. Disarmed reading allows the host to safely inspect a disarmed file to understand if it loads any shared libraries that lack the protection mechanisms of the application. We expand upon disarmed reading using two recent mitigation bypasses that highlight its necessity.

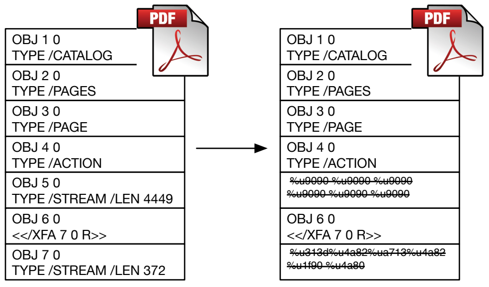
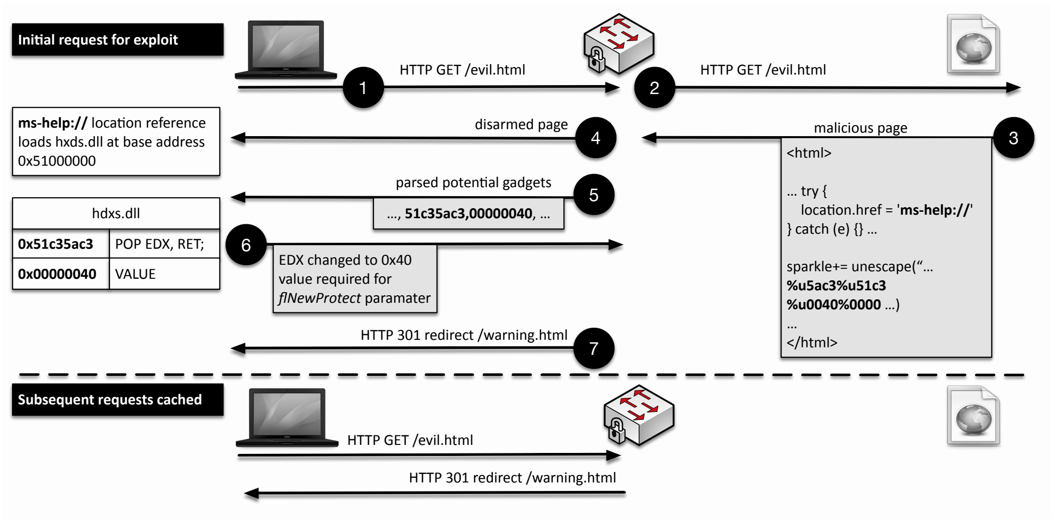


Figure 4. Disarmed reading of a malicious PDF

The first example considers the case where Adobe Reader suffers from a similar bypass technique. A properly crafted XFA tag within a PDF document can force the Adobe Reader application to load `icucnv36.dll`, which lacks ASLR [38]. For parameter suspicion to identify the gadget, it must be aware that the formatted file has loaded the additional shared library. Figure 4 depicts how disarmed reading handles loading a malicious PDF. The suspected file contains seven PDF objects: a catalogue, two pages, an action, two streams, and an object containing the XFA tag. Code-Stop removes the two streams when disarming the file, since streams prove to be common locations for ROP chains and shellcode. However, Code-Stop leaves the other objects intact. Object 6 0 contains the XFA tag that forces the Adobe Reader application to load `icucnv36.dll`. This results in disarmed reading learning of the base address of the shared library without ASLR. It further provides this information to parameter suspicion in order for it to have the full context of addresses that code exists at for the application under protection.

The second example considers the protection mechanisms of Internet Explorer, which can be bypassed by loading the `hxds.dll` by making a location reference to ms-help [22]. Because `hxds.dll` lacks ASLR, exploits can use fixed addresses within `hxds.dll` to construct an ROP chain capable of bypassing DEP. Parameter suspicion requires knowledge of what code exists at specific memory locations. Without knowing that an exploit forced Internet Explorer to load `hxds.dll` at the base address of 0x51BD0000, parameter suspicion cannot determine the emulated effect of any gadget. Ultimately, Code-Stop allows a disarmed version of an HTML document through the proxy such that `hxds.dll` loads, but does not contain any dynamic content that could be used to exploit the application.


Initial request for exploit
HTTP GET /evil.html
1
2
HTTP GET /evil.html
ms-help:// location reference loads hxds.dll at base address 0x51000000
disarmed page
4
malicious page
3
parsed potential gadgets
5
..., 51c35ac3,00000040, ...
hdxs.dll
0x51c35ac3 POP EDX, RET;
0x00000040 VALUE
6
EDX changed to 0x40 value required for flNewProtect paramater
HTTP 301 redirect /warning.html
7
Subsequent requests cached
HTTP GET /evil.html
HTTP 301 redirect /warning.html

```
<html>

... try {
   location.href = 'ms-help://'
} catch (e) {} ...

sparkle+= unescape("...
%u5ac3%u51c3
%u0040%0000 ...)
...
</html>
```

Figure 5. Example Detection of the Dynamically Loaded hxds.dll ROP Chain

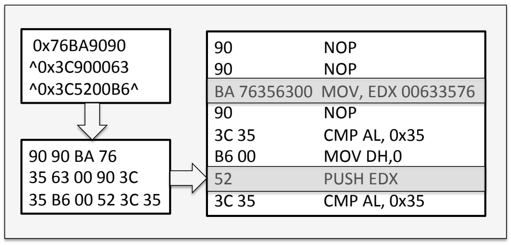
D. Detection of an Example Attack

Figure 5 depicts how our prototype prevents an example attack. (1) First, the victim requests the webpage evil.html that contains an exploit with a VirtualProtect() ROP chain that loads at runtime. (2) The proxy requests the webpage on behalf of the client and (3) receives the html page (4) The proxy then sends a disarmed form of the page to the client to determine what additional libraries might be loaded when the page loads. In this case, the location reference for ms-help:// loads a shared library without ASLR protection. (5) The proxy parses values that may be addresses of gadgets or part of a ROP Chain (e.g., - 0x51c35ac3,0x00000040). These potential gadgets are sent to the host to determine their context. (6) The host replies with the impact of the ROP Chain on the memory registers. In this case, we demonstrate the part of the ROP Chain that loads the value 0x40 into EDX as a parameter for VirtualProtect(). (7) After classifying the malicious impact of the ROP Chain, the proxy replies with a HTTP 301 redirect to a warning page. On subsequent requests - the proxy replies with the HTTP 301.

E. Extending Code-Stop

We now examine how the design of Code-Stop allows us to identify the distinct JOP code-reuse attack and other generic exploit behavior.

JOP Classifier: Jump-Oriented Programming (JOP) presents a unique code-reuse attack [39]. Instead of using gadgets ending in return instructions, JOP uses register-indirect jumps to chain together gadgets. JOP's design contains two types of gadgets: functional gadgets and dispatcher gadgets. Dispatcher gadgets essentially maintain a virtual program counter, advancing the attack and allowing functional gadgets to execute. A dispatcher gadget may prove as simple as `ADD EDX, 4; JMP [EDX]`, which repeatedly advances the virtual program counter by a constant value. Since dispatcher gadgets must alternate functional gadgets, we implement a classifier that identifies JOP similar to parameter suspicion. To detect JOP, Code-Stop identifies the alternating dispatcher gadgets by manipulation of a single register, and then a jump instruction. As with parameter suspicion, there is a negligible likelihood that a random string contains alternating addresses



```
0x76BA9090
^0x3C900063
^0x3C5200B6^

90 90 BA 76
35 63 00 90 3C
35 B6 00 52 3C 35

90           NOP
90           NOP
BA 76356300  MOV, EDX 00633576
90           NOP
3C 35        CMP AL, 0x35
B6 00        MOV DH,0
52           PUSH EDX
3C 35        CMP AL, 0x35
```

Figure 6. Identifying just-in-time code spraying

that point to instructions that happen to manipulate a single register and then jump to that register.

JIT Code-Spraying Classifier: We now discuss how to extend Code-Stop to detect other types of exploitation attack vectors, including JIT-Flash. JIT-Flash sprays executable code directly into memory. Consider the example depicted in Figure 6. A JIT-Flash attack writes a suspected string into memory. When Code-Stop translates the string to raw bytes, it replaces the XOR character with the ASCII encoding value 0x35. Next, we evaluate the bytes as variable-length x86 instructions. Emulating the instructions determines that the attack moves the value 0x00633576 into EDX and subsequently pushed that value onto the stack. To detect JIT Code-Spraying, we extend parameter suspicion to examine potential code and determine if the value has intentionally been placed onto the stack for malicious purposes. It is very unlikely that a benign arrangement of bytes would accomplish the same effect.

To identify JIT Code Spraying, Code-Stop examines potential JIT bytes to determine if they: 1) pop an address, 2) push an address to the flow of execution, or 3) store values at our heap-spray. In the case of Figure 6, the highlighted section of code clearly pushes an address to the flow of execution and is detected as an attack. Next, we describe our evaluation.

VI. Evaluation

We evaluate Code-Stop by answering the following research questions.

- **RQ1:** What is the accuracy of detecting gadgets?

- **RQ2:** What is the performance overhead on the client host?
- **RQ3:** What is the scalability of the network proxy?
- **RQ4:** What set of attacks can Code-Stop detect?

The following sections answers these questions and describes the configuration of our prototype used in the evaluation.

A. Experimental Setup

We tested our prototype using the following systems, which were configured as described below:

Network Proxy: *DansGuardian 2.10.1.1, Squid Version 3.3.8 on Ubuntu 14.04 LTS.* We implement our gadget parser as a DansGuardian content-scanner via a python script that parses suspect gadgets. The script communicates with the vulnerable host over TCP sockets to understand the effect of the emulation of the suspected gadgets.

Vulnerable Host: *Windows 7 Service Pack 1.* We installed applications that are known to be vulnerable, including Adobe Reader 9.0, Internet Explorer 8.0 and 11.0, JRE-1.6, and Microsoft Office 2010. It is necessary to test using these specific application versions to ensure we can properly test ROP chains from `icucnv36.dll`, `msvcr71.dll`, and `hxds.dll`. Additionally, we installed a third-party browser help object IBM Forms Viewer 4.0.0, which installed the `pehelper.dll` that is compiled without ASLR support. The host runs a Python script that determines the emulated impact of potential gadgets and communicates with the network proxy.

B. RQ1: Accuracy of Detecting Gadgets

The first part of our evaluation investigates the accuracy of our prototype to detect gadgets in PDF documents. We compare three cases: (a) matching a string that contains a hexadecimal address; (b) matching a string that contains a hexadecimal address corresponding to the address space of Adobe Acrobat Reader and its shared libraries; and (c) matching using parameter suspicion. In doing so, we show the benefit of Code-Stop over naive approaches.

Datatasets: We utilize the following datasets to illustrate the accuracy of our prototype.

- Contagio Datasets: The Contagio Benign Dataset consists of 9,000 known benign PDF documents from March 2013. The Contagio Malware Repository Team collected the dataset and published it for the specific purpose of testing security products for false positives. We used the dataset to ensure our prototype did not falsely detect benign documents as malicious in nature. In addition, the Contagio Team published a smaller dataset of 109 complex PDF documents that contained shockwave flash. This dataset was included for the specific testing of larger PDF files that contained large amounts of arbitrary data.
- VirusTotal Dataset: The VirusTotal Dataset includes 1,000 known benign documents from September 2015. The VirusTotal team made their private API available for testing our prototype. Using their private API, we queried for known benign documents that had been uploaded to their website within the last thirty days.
- Metasploit Dataset: We extended the Metasploit_adobe_toolbutton.rb exploit to include five additional ROP Chains for shared libraries without ASLR. We then randomly generated 550 unique malicious PDF documents with different payloads and different ROP Chains. Note that this is the only controlled dataset in our experiment evaluation as all PDFs contain known ROP chains. Using a controlled dataset, we can test Code-Stop's ability to detect code-reuse attacks since the application and operating system versions must match that of the attack. Alternatively, a wild dataset would encompass attacks against multiple versions of applications and operating systems.

Results: Table III depicts the results of executing the string matching algorithms on our dataset. The first row demonstrates that simply matching strings containing hexadecimal addresses produces a significant number of false positives. Refining the matching to values that are valid address ranges significantly reduces the number of false positives, but there are still some. However, using parameter suspicion's heuristic of identifying three register changes; we do not detect false positives.

C. RQ2: Host Performance Overhead

Next, we measured the performance overhead on the client to determine any negative impact when rendering content. We focused on JavaScript performance, since Code-Stop heavily parses and examines JavaScript for potential code-reuse attacks. We measured JavaScript performance using the Sun Spider JavaScript Benchmarking Suite [40]. Sun Spider measures the performance of the host executing core JavaScript language, focusing on the typical code implemented in real-world situations. We compared the Sun Spider results for the following scenarios: (a) a normal host (baseline), (b) a host using a Squid Proxy, (c) a host using the SquidClamAV Proxy with Avast anti-virus scanning, and (d) our Code-Stop solution. Sun Spider reports 95% confidence intervals as percentages.

Results: Table IV shows that Code-Stop suffers a minimal performance overhead penalty, comparable to the results of the SquidClamAV Proxy. Ultimately, the performance overhead is negligible, since the difference between the baseline and Code-Stop is less than 8ms, which is imperceptible to the end user.

D. RQ3: Network Proxy Scalability

In the previous experiment, we investigated the performance overhead on a client protected by Code-Stop. However, in a typical deployment, there will be many clients protected by the Code-Stop proxy. Therefore, it is important to characterize the scalability of Code-Stop to many clients. The experiment measured the average time to download a 10MB PDF file through the same four scenarios considered in Section VI-C.

We measured the average time to initiate and complete a download of the PDF given 1,5,10,15,20 and 25 concurrent users repeatedly downloading an uncached copy of the PDF over a period of 5 minutes. During the Code-Stop test, this resulted in the sum users downloading the 10MB file over 7,500 times over a period of 300 seconds.

Results: Figure 7 demonstrates that Code-Stop scales similar to a typical proxy configuration and an anti-virus solution. In an environment with 25 concurrent users, the time to download a 10MB document averaged 0.978 ± 0.129 seconds. In contrast, hosts under Code-Stop protection averaged 1.968 ± 0.262

TABLE III. CODE-STOP PARAMETER SUSPICION DETECTION IN BENIGN AND MALICIOUS DATASETS

	Total Files Tested	Total String Matches	Files with String Matches	Total Strings in Address Space	Files with Strings in Address Space	Files Matched by Parameter Suspicion
Contagio Benign Dataset	9,000	3,249,338	8,977	5,286	494	0
Contagio Complex Dataset	109	8,853	104	35	24	0
VirusTotal Benign Dataset	1,001	3,096,287	987	3,309	191	0
Metasploit Malicious Dataset	550	63,170	550	8,651	550	550

TABLE IV. RESULTS OF SUNSPIDER PERFORMANCE TEST

	Without Proxy	Standard Proxy	SquidClamAV Proxy	Code-Stop Proxy
Time (ms)	105.9ms ± 1.2%	106.5ms ± 1.2%	112.3ms ± 3.7%	113.8ms ± 2.8%

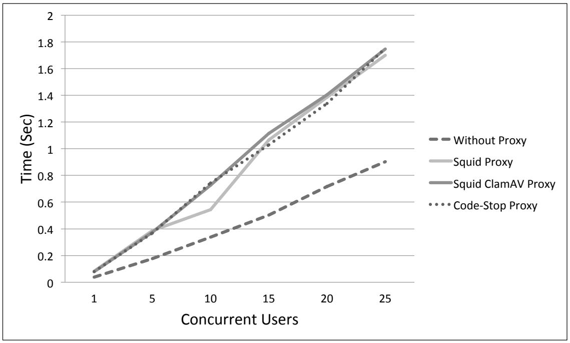


Figure 7. Performance impact with concurrent users

seconds. Both an anti-virus proxy and our prototype roughly double the time to complete the request to download a document. We argue that our solution proves superior to anti-virus scanning since the anti-virus scanning done on the proxy cannot take into account the full context of the host.

E. RQ4: Coverage

Finally, we evaluated the coverage of our Code-Stop prototype. Coverage defines the set of attacks our system can detect under ideal conditions. We evaluated parameter suspicion alone, as well as its use in combination with disarmed reading to detect code-reuse attacks that rely upon loading shared libraries without protection. Further, we measured Code-Stop against a set of five unique shared libraries that lack ASLR. We argue that Code-Stop is not limited to the method, the library, or the individual gadgets used in the code-reuse attack but is a technique that is applicable to a broader set of attacks.

Dataset: We evaluated how parameter suspicion detected ROP chains from different shared libraries. Specifically, we sought libraries that had one or more of the following characters: (1) compiled without ASLR (2) used with multiple different applications (3) contains the necessary instructions to generate an ROP chain, and (4) has been seen in use in the wild in attacks. Thus we selected the following libraries `icucnv36.dll`, `cryptocme2.dll`, `grooveutil.dll`, `pehelper.dll`, and `msvcr71.dll` respectively. Section VI-B previously demonstrated that our prototype detected malicious PDFs with ROP chains from the first three libraries with a malicious PDF. To further demonstrate the coverage, we modified the script for the Metasploit ie_cgenericelement_uaf.rb exploit for Internet Explorer to include new ROP chains from `pehelper.dll`, `msvcr71.dll`, and `grooveutil.dll`. In doing this, we demonstrate that Code-Stop is applicable to a broader set of applications since the technique detects the general behavior of code-reuse attacks and is not unique to any one application or shared library.

Results: Code-Stop detected the gadgets from the shared libraries of `pehelper.dll`, `msvcr71.dll`, and `grooveutil.dll`. Since these libraries are from a third-party application, the operating system, and a separate Microsoft application we can argue that the results indicate that parameter-suspicion covers the broad spectrum of shared libraries used to implement code-reuse attacks.

VII. Limitations and Future Work

Limitations of 32-Bit Architecture: For simplicity, we implemented the Code-Stop prototype against the 32-bit architecture. Extending Code-Stop to 64-bit increases the accuracy of the system, by decreasing the probability that a random string would refer to an address in the 64-bit address space.

Limitations of De-obfuscation: Additionally, we do not address obfuscation in the design of Code-Stop. Other works have examined the de-obfuscation of malicious JavaScript, PDF document streams, or Adobe Flash ActionScript [32] [41]. Implemented as a proxy, Code-Stop can prevent access to files it is unable to de-obfuscate. Future work may examine the concept of a quarantined machine that can emulate the full effect the result of rendering the obfuscated file and determine if the file attempts to allocate potential gadgets into memory.

Extending to Other Operating Systems: Further, we implemented the Code-Stop host software only as a Microsoft Windows application. Extending the host software to other platforms allows the possibility to protect against platform-unique attacks, such as S-ROP. In this paper, we addressed classifiers for generic code-reuse attacks and specific cases for ROP and JOP. Future work may extend classifiers to identify other attack vectors outside of the scope of code-reuse attacks as demonstrated with our JIT Code-Spraying classifier.

Allowing the Host to Proxy Content: Last, we implemented the gadget parser on the proxy and rely on the host to deliver context. Both could be implemented on the host. Splitting the design allows both the network and the host to share the performance overhead and prevents the attacker from opting out of both defense mechanisms. Future work may examine the host both parsing gadgets and examining context.

VIII. Related Work

In the following section, we describe the related work that addresses the shortcomings of defense strategies against code-reuse attacks. First, we examine the recent work into mitigating code-reuse using compiler- and operating system- based mechanisms. Next, we examine defenses utilizing the network layer. We believe our solution can combine the benefits of both the host and network layer defenses to mitigate code-reuse attacks.

A. Host-Level Code-Reuse Prevention

Because of ROP's success in bypassing DEP, several papers have examined means for defending against ROP [4] [42] [43]. Onarlioglu et al. [5] presented G-Free to counter ROP gadgets by removing unaligned gadgets altogether, and by removing a portion of aligned gadgets from binaries at compile-time. Alignment checking proved to be one of the more trivial but successful mechanisms for defeating ROP gadgets. Gadgets can consist of different offsets inside valid instructions. G-Free extends GCC to ensure a gadget-free binary. However, G-Free fails to protect already compiled binaries or shared libraries.

Other mechanisms such as kBouncer, ROPGuard, and ROPecker use the processor's Last Branch Recording (LBR) functionality to heuristically examine control flow for gadgets [25]. kBouncer [26] examines the LBR for fifty-two WinAPI functions considered harmful. RopGuard [7] expands upon this by performing past and future control flow analysis and static checks. By preceding protected API calls, it ensures that an attacker cannot divert into a protected function. Furthermore, RopGuard performs checks to ensure a process does not attempt to make the stack executable by disabling DEP. ROPecker [8] attempts to combine the benefits of RopGuard and kBouncer by examining the LBR control-flow simulation. However, the LBR defenses have proven trivial to bypass by clearing the LBR. Schustere et al. [25] demonstrated that both i-long-jumps and LBR flushing gadgets can defeat the effectiveness of any mitigation strategy that relies on the LBR.

Several recent defenses have prevented memory disclosures that could be used to create gadgets just-in-time (JIT). HideM [44] uses a split translation look-aside buffer to fetch read and executable memory separately. However, Crane et al. [45] noted that the split TLB technique does not work on recent x86 processors, since most processors released after 2008 contain a unified second level TLB. In contrast, Crane et al. [45] presented Redactor that successfully disassembles code pages and identifies JIT-ROP gadgets dynamically at runtime. Additionally, De Groef et al. [9] presented a countermeasure for JIT-ROP gadgets. When faced with the difficult problem that JIT gadgets can bypass ASLR and W$\oplus$X policies, De Groef and his team implemented a run-time monitor to prevent JIT ROP. The monitor uses a series of checks against any system call generated from the stack or heap, in order to determine the original calling function. However, DeMott [3] demonstrated bypassing caller check monitors by borrowing from valid code that makes calls to protected APIs.

B. Network-Level Exploit Prevention

Several checks can be performed at the network-layer to prevent the successful exploitation of hosts. Early, signature-based checks identified specific x86 instructions commonly used in malicious shellcode. However, the vast abundance of code-obfuscation techniques made these early checks easy for an attacker to bypass. One of the earlier methods for detection improved on such techniques by using a NIDS-embedded CPU emulator [27]. This emulator executed potential instructions with the intent of identifying polymorphic shellcode that evaded signature-based detection. In a similar approach, SigFree [28] presented a model for implementing a proxy-based firewall. This firewall successfully identified and filtered client-side exploits by detecting code and examining instructions using a process of code abstraction. With the extensive amount of client-side code executing in the context of the modern browser, the 2006 results appear to be only applicable in theory.

In 2010, researchers developed the JSAND toolkit [29] to emulate JavaScript and reliably identify malicious code based on machine-learning. However, this approach is limited to JavaScript and does not affect the large volume of exploits delivered via modern plug-ins, such as Adobe Flash. Support Vector Machines (SVMs) shellcode detection egingines use modern emulation to identify key instructions commonly used in shellcode [46]. Polychronakis and Keromytis [24] specifically proposed a network tool that could identify potential ROP Gadgets, but did so without dynamic context from the host. We argue that the shortcomings of the host-layer and network-layer defenses can be addressed by using the network to defend with the context of the host.

IX. Conclusion

This paper presented Code-Stop, a network and host-based cooperative system for defending against client-side attacks that bypass exploit mitigation strategies using code-reuse. We introduced the concept of *parameter suspicion* classifier to identify code-reuse attacks. With parameter suspicion, we introduced the idea of identifying when an attacker attempts to call a Windows API function with specific parameters to allocate memory or change memory protections. Rather than identifying the specific call to a particular function, parameter suspicion identifies code-reuse attacks by emulating the behavior of suspected instructions, gained from the context of the destination host. Parameter suspicion identifies when instructions place parameters onto the stack as part of a code-reuse attack to call a Windows API function to allocate or change memory. By examining the host context, Code-Stop can effectively prevent client-side attacks from reaching the intended victim. We implemented a prototype of Code-Stop, and verified the ability to mitigate exploits against Internet Explorer, Adobe Reader, and Microsoft Office applications. Our evaluation showed that Code-Stop successfully prevented code-reuse attacks without risk of false-positives. With the ability to detect code-reuse attacks and mitigate against previously unseen attack vectors, Code-Stop is a practical solution for enhancing the security of client-side applications.

References

[1] Microsoft Corporation, "BlueHat Prize Winners Announced," Aug 2012, Retrieved: April 10, 2016. [Online]. Available: http://www.microsoft.com/security/bluehatprize/

[2] H. Shacham et al., "On the Effectiveness of Address-Space Randomization," in Proc. of the ACM conference on Computer and communications security, Oct 2004, pp. 298–307.

[3] J. DeMott, "Bypassing EMET 4.1," Bromium Labs, Feb 2014, Retrieved: April 10, 2016. [Online]. Available: http://labs.bromium.com/2014/02/24/bypassing-emet-4-1/

[4] V. Pappas, M. Polychronakis, and A. D. Keromytis, "Smashing the Gadgets: Hindering Return-Oriented Programming Using In-Place Code Randomization," in Proc. of the IEEE Symposium on Security and Privacy, 2012, pp. 601–615.

[5] K. Onarlioglu, L. Bilge, A. Lanzi, D. Balzarotti, and E. Kirda, "G-Free: Defeating Return-oriented Programming Through Gadget-less Binaries," in Proc. of the Annual Computer Security Applications Conference (ACSAC), 2010, pp. 49–58.

[6] V. Pappas, M. Polychronakis, and A. D. Keromytis, “Transparent ROP Exploit Mitigation Using Indirect Branch Tracing,” in Proc. of the USENIX Conference on Security, 2013, pp. 447–462.

[7] I. Fratric, “Runtime Prevention of Return-Oriented Programming Attacks,” in Microsoft’s BlueHat Prize - Black Hat USA 2012. June, 2012. [Online]. Available: https://github.com/ivanfratric/ropguard

[8] Y. Cheng, Z. Zhou, M. Yu, X. Ding, and R. H. Deng, “ROPecker: A generic and practical approach for defending against ROP attacks,” in Proc. of the Symposium on Network and Distributed System Security (NDSS), 2014, pp. 763–780.

[9] W. De Groef, N. Nikiforakis, Y. Younan, and F. Piessens, “Jitsec: Just-in-time security for code injection attacks,” in Benelux Workshop on Information and System Security (WISSEC), Nov 2010, pp. 1–15.

[10] J. C. Chen and B. Li, “Evolution of exploit kits : Exploring past trends and current improvements,” Trend Micro, Tech. Rep., 2015.

[11] S. Bhatkar, D. C. DuVarney, and R. Sekar, “Address Obfuscation: An Efficient Approach to Combat a Broad Range of Memory Error Exploits,” in Proc. of the 12th USENIX security symposium, vol. 120. Washington, DC., Aug 2003, pp. 105–120.

[12] PaX Team, “Pax address space layout randomization (aslr),” Mar 2003, Retrieved: April 10, 2016. [Online]. Available: http://pax.grsecurity.net/docs/aslr.txt

[13] C. Kil, J. Jun, C. Bookholt, J. Xu, and P. Ning, “Address Space Layout Permutation (ASLP): Towards Fine-Grained Randomization of Commodity Software,” in Proc. of the Annual Computer Security Applications Conference (ACSAC), 2006, pp. 339–348.

[14] A. Sotirov and M. Dowd, “Bypassing browser memory protections in windows vista,” in Blackhat USA, Aug 2008.

[15] T. Haq, “Internet explorer 8 exploit found in watering hole campaign targeting chinese dissidents,” Mar 2013, Retrieved: April 10, 2016. [Online]. Available: https://www.fireeye.com/blog/threat-research/2013/03/internet-explorer-8-exploit-found-in-watering-hole-campaign-targeting-chinese-dissidents.html

[16] T. OConnor, “Nuclear Scientists, Pandas and EMET Keeping Me Honest,” SANS Internet Storm Center, Tech. Rep., May 2013, Retrieved: April 10, 2016.

[17] D. Litchfield, “Buffer Underruns, DEP, ASLR and improving the Exploitation Prevention Mechanisms (XPMs) on the Windows platform,” in Next Generation Security Software, Sep 2005.

[18] E. J. Schwartz, T. Avgerinos, and D. Brumley, “Q: Exploit Hardening Made Easy,” in Proc. of the USENIX Security Symposium, Aug 2011, pp. 25–41.

[19] P. Team, “Pax non-executable pages design & implementation,” May 2003, Retrieved: April 10, 2016. [Online]. Available: http://pax.grsecurity.net/docs/noexec.txt

[20] H. Shacham, “The Geometry of Innocent Flesh on the Bone: Return-into-libc Without Function Calls (on the x86),” in Proc. of the ACM conference on Computer and communications security, Oct 2007, pp. 552–561.

[21] D. Dai Zovi, “Practical return-oriented programming,” 2010, Retrieved: April 10, 2016. [Online]. Available: https://www.trailofbits.com/resources/practical_rop_slides.pdf

[22] P. Anwar, “Bypassing Microsoft Windows ASLR with a little help by MS-Help,” Aug 2012, Retrieved: April 10, 2016. [Online]. Available: https://www.greyhathacker.net/?p=585

[23] N. Sikka, “Cve 2013 3893: Fix it workaround available,” Microsoft Corporation, Tech. Rep., Sep 2013, Retrieved: April 10, 2016. [Online]. Available: http://blogs.technet.com/b/srd/archive/2013/09/17/cve-2013-3893-fix-it-workaround-available.aspx

[24] M. Polychronakis and A. D. Keromytis, “ROP Payload Detection Using Speculative Code Execution,” in Proc. of the International Conference on Malicious and Unwanted Software (MALWARE), 2011, pp. 58–65.

[25] F. Schuster et al., “Evaluating the Effectiveness of Current Anti-ROP Defenses,” in Research in Attacks, Intrusions and Defenses, vol. 8688, 2014, pp. 88–108.

[26] L. Davi, P. Koeberl, and A.-R. Sadeghi, “Hardware-Assisted Fine-Grained Control-Flow Integrity: Towards Efficient Protection of Embedded Systems Against Software Exploitation,” in Proc. of the Annual Design Automation Conference, 2014, pp. 1–6.

[27] M. Polychronakis, K. G. Anagnostakis, and E. P. Markatos, “Network-Level Polymorphic Shellcode Detection Using Emulation,” in Proc. of Detection of Intrusions and Malware & Vulnerability Assessment. Springer, Jul 2006, pp. 21:1–21:3.

[28] X. Wang, C.-C. Pan, P. Liu, and S. Zhu, “Sigfree: A Signature-Free Buffer Overflow Attack Blocker,” IEEE Transactions on Dependable and Secure Computing, vol. 7, Jun 2010.

[29] M. Cova, C. Kruegel, and G. Vigna, “Detection and Analysis of Drive-by-download Attacks and Malicious JavaScript Code,” in Proc. of the International Conference on World Wide Web (WWW), 2010.

[30] Z. Tzermias, G. Sykiotakis, M. Polychronakis, and E. P. Markatos, “Combining static and dynamic analysis for the detection of malicious documents,” in Proc. of the Fourth European Workshop on System Security. ACM, 2011, p. 4.

[31] D. Stevens, “Malicious pdf documents explained,” Security & Privacy, IEEE, vol. 9, no. 1, 2011.

[32] B. Hartstein, “Jsunpack: An automatic javascript unpacker,” in ShmooCon convention, 2009.

[33] N. Šrndic and P. Laskov, “Detection of malicious pdf files based on hierarchical document structure,” in Proc. of the 20th Annual Network & Distributed System Security Symposium. Citeseer, 2013.

[34] M. Egele, P. Wurzinger, C. Kruegel, and E. Kirda, “Defending browsers against drive-by downloads: Mitigating heap-spraying code injection attacks,” in Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, 2009, pp. 88–106.

[35] Y. Yu, “Rops are for the 99 percent,” Mar 2014, Retrieved: April 10, 2016. [Online]. Available: https://cansecwest.com/slides/2014/ROPs_are_for_the_99_CanSecWest_2014.pdf

[36] M. S. Xiaobo Chen, Dan Caselden, “New zero-day exploit targeting internet explorer versions 9 through 11 identified in targeted attacks,” Apr 2014, Retrieved: April 10, 2016. [Online]. Available: https://www.fireeye.com/blog/threat-research/2014/04/new-zero-day-exploit-targeting-internet-explorer-versions-9-through-11-identified-in-targeted-attacks.html

[37] P. V. Eeckhoutte, “Corelan team exploit development swiss army knife,” 2015, Retrieved: April 10, 2016. [Online]. Available: https://github.com/corelan/mona/blob/master/mona.py

[38] J. V. Soroush Dalili, sinn3r, “Adobe reader toolbutton use after free,” Nov 2013, Retrieved: April 10, 2016. [Online]. Available: http://www.rapid7.com/db/modules/exploit/windows/browser/adobe_toolbutton

[39] T. Bletsch, X. Jiang, V. W. Freeh, and Z. Liang, “Jump-oriented Programming: A New Class of Code-reuse Attack,” in Proc. of the ACM Symposium on Information, Computer and Communications Security (ASIACCS), 2011, pp. 30–40.

[40] J. K. Martinsen, H. Grahn, and A. Isberg, “A Comparative Evaluation of JavaScript Execution Behavior,” in Proc. of the International Conference on Web Engineering, 2011, pp. 399–402.

[41] J. M. Esparza, “Obfuscation and (Non-) Detection of Malicious PDF Files,” in CARO 2011 Workshop in Prague, Czech Republic, 2011.

[42] L. Davi, A.-R. Sadeghi, D. Lehmann, and F. Monrose, “Stitching the Gadgets: On the Ineffectiveness of Coarse-grained Control-flow Integrity Protection,” in Proc. of the USENIX Conference on Security Symposium, 2014, pp. 01–416.

[43] L. Davi, A.-R. Sadeghi, and M. Winandy, “ROPdefender: A Detection Tool to Defend Against Return-oriented Programming Attacks,” in Proc. of the ACM Symposium on Information, Computer and Communications Security, 2011, pp. 40–51.

[44] J. Gionta, W. Enck, and P. Ning, “Hidem: Protecting the contents of userspace memory in the face of disclosure vulnerabilities,” in Proc. of the ACM Conference on Data and Application Security and Privacy (CODASPY), 2015, pp. 325–336.

[45] S. Crane and et al., “Readactor: Practical Code Randomization Resilient to Memory Disclosure,” in Proc. of the IEEE Symposium on Security and Privacy, 2015, pp. 763–780.

[46] Y. Hou, J. W. Zhuge, D. Xin, and W. Feng, “SBE : A Precise Shellcode Detection Engine Based on Emulation and Support Vector Machine,” in Proc. of the International Conf. on Information Security Practice and Experience, 2014, pp. 159–171.

Detecting Obfuscated JavaScripts using Machine Learning

Simon Aebersold*, Krzysztof Kryszczuk*, Sergio Paganoni†, Bernhard Tellenbach*, Timothy Trowbridge*
*Zurich University of Applied Sciences, Switzerland
†GovCERT.ch, Reporting and Analysis Centre for Information Assurance MELANI

***Abstract*—JavaScript is a common attack vector for attacking browsers, browser plug-ins, email clients and other JavaScript enabled applications. Malicious JavaScripts redirect victims to exploit kits, probe for known vulnerabilities to select a fitting exploit or manipulate the Document Object Model (DOM) of a web page in a harmful way. Malicious JavaScript code is often obfuscated in order to make it hard to detect using signature-based approaches. Since the only other reason to use obfuscation is to protect intellectual property, the share of scripts which are both benign and obfuscated is quite low, and could easily be captured with a whitelist. A detector that can reliably detect obfuscated JavaScripts would therefore be a valuable tool in fighting malicious JavaScripts. In this paper, we present a method for automatic detection of obfuscated JavaScript using a machine-learning approach. Using a dataset of regular, minified and obfuscated samples from a content delivery network and the Alexa top 500 websites, we show that it is possible to distinguish between obfuscated and non-obfuscated scripts with precision and recall around 99%. We also introduce a novel set of features, which help detect obfuscation in JavaScripts. Our results presented here shed additional light on the problem of distinguishing between malicious and benign scripts.**

***Index Terms*—Computer security; Machine learning; Pattern analysis; Classification algorithms; JavaScript, Random Forest; Malicious**

I. Introduction

JavaScript is omnipresent on the web. Almost all websites make use of it and there are a lot of other applications, such as Portable Document Format (PDF) forms or HyperText Markup Language (HTML) e-mails, where JavaScript plays an important role. This strong dependence creates an attack opportunity for individuals looking for an entry point into a victims system. The main functionalities of a malicious JavaScript are reconnaissance and exploitation, and cross-site scripting (XSS) vulnerabilities in web applications.

JavaScript exploit kits belong to the first category of functionality and typically contain code for identification of the victim's browser and its plug-ins. Most of the malicious JavaScripts are obfuscated in order to hide what they are doing and to evade detection by signature based security systems. Since the only other reason to use obfuscation is to protect intellectual property, the share of benign obfuscated scripts is quite low and could probably be captured with a whitelist. A detector that can reliably detect obfuscated JavaScripts would therefore be a valuable tool in fighting malicious JavaScripts.

The most common method to address the problem of malicious JavaScripts is having malware analysts write rules for anti-virus or intrusion detection systems that identify common patterns in obfuscated (or non-obfuscated) malicious scripts. While signature based detection is good at detecting known malware, it often fails to detect it when obfuscation is used to alter the features captured by the signature [1]. Furthermore, keeping up with the attackers and their obfuscation techniques is a time consuming task. This is why a lot of research effort is put into alternative solutions to identify/classify malicious JavaScripts. One area is to automate at least parts of the manual analysis required to identify whether or not a script is malicious and to craft suitable signatures. JSDetox [2] and Wepawet [3] are two solutions that help with the dynamic analysis of JavaScript samples.

Likarish *et al.* [4] take another approach and apply machine learning algorithms to detect obfuscated malicious JavaScript samples. The authors use a set of 15 features like the number of strings in the script or the percentage of white-space that are largely independent from the language and JavaScript semantics. The results from their comparison of four machine learning classifiers (naive bays, ADTree, SVM and RIPPER) are very promising: the precision and recall (see III-E for a definition) of the SVM classifier is 92% and 74.2%. But since their study originates from 2009, it is unclear how recent trends like the minification of JavaScripts (see II-A) would impact on their results.

A more recent study from Kaplan *et al.* [5] addresses the problem of detecting obfuscated scripts using a Bayesian classifier. They refute the assumption made by previous publications that obfuscated scripts are mostly malicious and advertise their solution as filter for projects where users can submit applications to a software repository such as a browser extension gallery for browsers like Google Chrome or Firefox. Also techniques such as AdSafe [6], severely restrict what is allowed JavaScript and what not to simplify analysis.

Wang *et al.* [7] propose another machine learning based solution to separate malicious and benign JavaScript. They compare the performance of ADTree, NaiveBayes and SVM machine learning classifiers using a set of 27 features of which some are similar to those of Likarish *et al.* [4]. Their results suggest a significant improvement over the work of Likarish *et al.*

In this paper, we present a method for automatic detection of obfuscated JavaScript using a machine-learning approach. We confirm results from other researchers that using approaches based on machine learning, it is possible to distinguish between obfuscated and non-obfuscated scripts with precision and recall above 95%. Our results complement previous research in that they expose a substantial challenge to obtain those good results. Our results suggest that it is difficult to train detectors to be robust versus changes in the way obfuscation is done. If there are no samples of scripts obfuscated with a specific obfuscation tool or method, detection rates drop significantly.

Today's JavaScript is mostly minified code. The second contribution of this paper is an investigation whether minification has an impacts on the detection of obfuscated JS using machine learning techniques. Finally, we shed additional light on the problem of distinguishing between malicious and benign scripts using a custom database. In our experiments we could not confirm the promising results from previously published research, where similar features and machine learning approach was taken.

The rest of the paper is organized as follows. Section II briefly explains the different JavaScript classes, which we aimed to separate in this work. In Section III, we discuss the machine-learning approach adopted in the presented work, as well as the discriminatory features and classifiers we used. Section IV presents our results, followed by a discussion and conclusions in Section V.

II. Syntactic and functional varieties of JavaScript

The client-side JavaScript for JavaScript-enabled applications can be attributed to one of the following four classes: regular, minified, obfuscated and malicious. The regular class contains the scripts as they have been written by their developers. These scripts are typically easy to read and understand by human beings. Obfuscation and minification are code modifications that change the syntax but not the functionality of the code. In this work, we refer to different syntactic and functional varieties of JavaScript as *classes*.

A. Minification

Since the introduction of the YUI Compressor [8] and other minification tools, more and more JavaScript in the Internet is minified. It is considered good practice to concatenate and minify JavaScript files to arrive at smaller file sizes and fewer requests. Minification removes spaces, line breaks and renames functions and variables to obtain a more compact version of the script. While this makes the scripts harder to read and understand for a human, the program flow remains the same.

B. Obfuscation

In contrast to minifiers, obfuscation tools do modify the program flow with the goal to make it hard to understand while keeping the original functionality. Many obfuscation techniques exist. For example, encoding obfuscation encodes strings using hexadecimal character encoding or Unicode encoding to make strings harder to read. Other obfuscation steps involve hiding code in data to execute it later using the eval JavaScript function. The following listing shows a sample use of the latter technique:

```
var a = "ale";
a += "rt(";
a += "'hello'";
a += ");";
eval(a);
```

Listing 1. A simple example for data obfuscation

C. Malicious vs benign

The dichotomy benign/malicious is of functional rather than syntactic nature. In contrast to the regular, minified and obfuscated class, scripts in the malicious class can have a regular form or make use of minifiers or obfuscators. This makes it difficult to detect those scripts using features that focus on differentiating the first three classes only. Previous work sometimes conflates obfuscation with maliciousness. In this work and in prior art (see [5]), it is explicitly stated that neither all obfuscated code is malicious nor is all malicious code obfuscated. Although formally speaking malicious JavaScript does not have to be obfuscated, in practice, it usually is.

III. Machine learning approach to JavaScript classification

In order to evaluate the feasibility and accuracy of distinguishing between different classes of JavaScript, we adopted a classical machine learning approach. We collected a database containing a number of instances representing each of the classes of interest, i.e., regular, minified, obfuscated, benign and malicious. For each of the samples in the database we extracted a set of discriminatory features, which we list in Table II below. The extracted features form fixed-length feature vectors, which in turn are used for training and evaluation of classifiers.

A. Data Set

Our dataset consists of data from three different sources: (1) the complete list of JavaScripts available from the *jsDelivr* content delivery network, (2) the *Alexa Top 500* websites and (3) a set of malicious JavaScript samples from the Swiss Reporting and Analysis Centre for Information Assurance *MELANI*.

jsDeliver: contains a large number of JavaScript libraries and files in both a regular and a minified version. Since the files are subject to manual review and approval and should not make use of obfuscation, we used the regular versions of the files as a basis for our evaluation. After a preprocessing step including rule-based filtering, de-duplication as well as manual sampling to check and make sure that the assumed properties (minified, obfuscated or malicious) are met, we generated three additional file sets from these files. For the first set, we processed the files with uglifyjs [9], the most popular JavaScript minifier to obtain a minified version of them. uglifyjs works by extracting an abstract syntax tree (AST) from the JavaScript source and then transforming it to an optimized (smaller) one. For the second and third set, we used the Dean Edwards' Packer [10] and javascriptobfuscator.com [11] to create obfuscated versions of these files. Note that the second and third set can also be considered to be minified. The two obfuscators remove whitespaces and make the scripts more compact. Scripts that are first minified and then obfuscated look similar or are the same as when obfuscation is applied only. Applying obfuscation and then minification might lead to partial de-obfuscation (e.g., decoding of encoded strings) and is therefor unlikely to be used in practice.

Alexa Top 500: To have a more comprehensible representation of actual scripts found on websites [12], we created a set of files consisting of the JavaScripts found on the Alexa Top 500 websites [13]. To extract the scripts from these websites, we parsed them with BeautifulSoup [14] and extracted all scripts that were either inlined (e.g., `<script>alert("foo");</script>`) or referenced via external files (e.g., `<script type="text/javascript" src="filename.js" ></script>`). Since we make no assumption about the properties of these files other than that they are non-malicious, no preprocessing was performed except for de-duplication.

MELANI: The fileset from MELANI contains only malicious samples. Most of the malicious samples in the set are either JS droppers used in drive-by-download attacks or Exploit Kits for exploiting vulnerabilities in browser plugins. Most samples are at least partially obfuscated and seem to make use of different obfuscation techniques and tools.

B. Preprocessing

Since a manual inspection of a random subset of the non-minified files downloaded from jsDeliver showed that 10% of them were minified nevertheless, we preprocessed them by applying the following simple heuristic:

- Remove files with less than 5 lines
- Remove files if less than 1% of all characters are spaces.
- Remove files where more than 10% of all lines are longer than 1000 characters).

While we did not inspect all of the removed files (around 10% of the files), a manual inspection of a subset of them showed no false positives. We did not find false negatives in the files that were not removed. It must be noted, however, that the above heuristic may not necessary be valid for other JavaScript datasets. If the data source contains a large number of small JavaScript snippets, the first rule might prove problematic.

In a next step, we used DoubleKiller [15] to remove all duplicate files. DoubleKiller compares files based on file name, size, modification date and content (CRC32). After preprocessing and de-duplication of the jsDeliver fileset a total of 4218 unique JavaScript files were left. After de-duplication of the Alexa Top 500 fileset, a total of 9459 files remained.

C. Feature Selection

For our experiments reported in this paper, we selected a set of 21 features derived from manual inspection, related work ([4], [16]) and analysis of the histograms of candidate features. For example, observations showed that obfuscated scripts often make use of encodings using hexadecimal or Unicode characters (F17) and often remove white spaces (F8). Furthermore, some rely on splitting a job in a lot of functions (F14) and almost all use a lot of strings (F7) and are lacking comments (F9). An example of a comparison of feature distributions across classes is shown in Figure 1. Here, it can be noted that if a script has 70% or more of its characters in strings, this is a strong indication that the file is obfuscated or malicious.

TABLE I
DATA COLLECTIONS

Collection	Properties	#Files
jsDelivr.com	regular	4218
jsDelivr.com	minified (uglifyjs)	4218
jsDelivr.com	obfuscated (Dean Edwards Packer)	4218
jsDelivr.com	obfuscated (javascriptobfuscator.com)	4218
Alexa Top 500	unknown	9459
MELANI	malicious, obfuscated	132

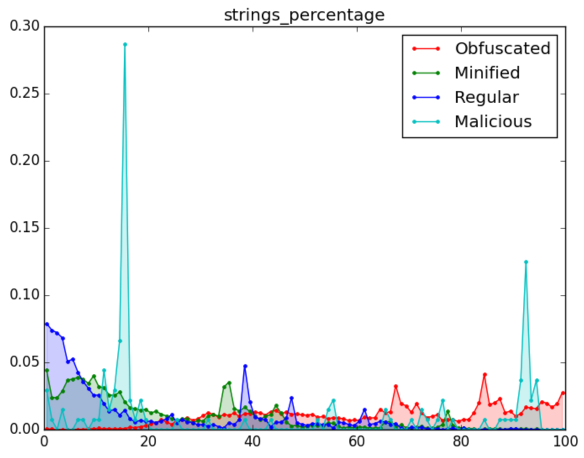


Fig. 1. Histogram for feature F7: The share of scripts that have x% of their characters in strings.

Table II lists the discriminatory features we used for training and evaluation of the classifiers in the reported experiments. These features are complemented with 26 features reflecting the frequency of 26 different JavaScript keywords: break, case, catch, class, continue, do, else, false, finally, for, if, instanceof, new, null, return, switch, this, throw, true, try, typeof, var, while, toString, valueOf and undefined. While the present set yielded promising results in our experiments, further investigations are required to determine an optimal set of classification features for the problem. The features labeled as 'new' in Table II are a novel contribution of the present paper. The special JavaScript elements used in feature F15 are elements often used and renamed (to conceal their use) in obfuscated or malicious scripts. This includes the following functions, objects and prototypes:

- **Functions**: eval, unescape, String.fromCharCode, String.charCodeAt
- **Objects**: window, document
- **Prototypes**: string, array

TABLE II
DISCRIMINATORY FEATURES

Feature	Description	Used in:
F1	total number of lines	[4]
F2	avg. # of chars per line	[4]
F3	# chars in script	[4]
F4	% of lines ¿1000 chars	new
F5	Shannon entropy of the file	[16]
F6	avg. string length	[4]
F7	share of chars belonging to a string	new
F8	share of space characters	[4]
F9	share of chars belonging to a comment	[4]
F10	# of `eval` calls divided by F3	new
F11	avg. # of chars per function body	new
F12	share of chars belonging to a function body	new
F13	avg. # of arguments per function	[4]
F14	# of function definitions divided by F3	new
F15	# of special JavaScript elements divided by F3	new
F16	# of renamed special JavaScript elements divided by F3	new
F17	share of encoded characters (for example `\u0123` or `\x61`)	[4]
F18	share of backslash characters	new
F19	share of pipe characters	new
F20	# of array accesses using dot or bracket syntax divided by F3	new

D. Feature Extraction

To extract the above features, we implemented a Node.js application traversing the abstract syntax tree (AST) generated by Esprima [17], a JavaScript parser compatible with Mozilla's SpiderMonkey Parser API [18]. Traversing the AST and extracting all of the above features for an average JavaScript library with around 20K characters takes around 330ms.

E. Machine Learning

The extracted set of feature vectors was used to train and evaluate three different classifiers:

- **Linear Discriminant Analysis (LDA)** Due to it's simple design it avoids function overfitting. This means it is still possible to overtrain this classifier, but only by training insufficient amount of data.
- **Random Forest (RF)** Random Forest uses a tree based approach. In comparison to decision trees it is less prone to overfitting because it selects a random subset of features to build multiple decision tress.
- **Support Vector Machine (SVM)** SVM works by searching a hyperplane in a feature space that separates labels/-classes. We use a radial basis function kernel (RBF) with parameters γ=0.03, C=8.0 to capture non linearities.

For exhaustive details on the used classifiers, the reader is referred to [19]. We used scikit-learn [20], which contains the implementation of the above mentioned classifiers. We performed following steps per experiment:

1) Normalization of the data using the StandardScaler of scikit-learn
2) Random partitioning of the data set to be evaluated into training and testing subsets. With one exception (see IV-A), 60% of the data are used for training, 40% for testing.
3) Training and testing of the classifiers. The partitioning and the training and testing is done 10 times (10-fold cross validation).
4) The results from the 10 rounds are averaged and the standard deviation is calculated.

We report the (p)recision, (r)ecall, (f1)-score and (s)upport for each considered class and considered classifier. Precision is the number of true positives divided by the number of true positives and false positives. Recall is the number of true positives divided by the number of true positives and the number of false negatives. The F1 Score conveys the balance between precision and recall and is equal to $2*\frac{precision*recall}{precision+recall}$. Finally, support is the total number of scripts tested for a specific label. Note that since we used 10-fold cross-validation, this number is the sum of the scripts tested in the 10 runs.

IV. RESULTS

In this section, we present how the partitioning of the dataset impacts on classification accuracy and how well the classifiers are able to distinguish obfuscated from non-obfuscated scripts. Finally, we show our results from experiments where the classifiers had to distinguish malicious from benign samples.

A. Partitioning and Accuracy

To check the impact of the size of the training set on the observed accuracy, we first tested all splits from 1% to 99% of test versus training data and calculated the share of scripts whose label is predicted correctly. Figure 2 shows the impact of the split of training and test data on the observed accuracy. The x-axis shows the test dataset size in %, the y-axis shows the accuracy in %. The training dataset size is equal to 100% minus the test dataset size. Over 95% of the scripts are labeled correctly for all classifiers if at least 15% of the data vectors is used for training. At 60% training data, 2 out of 3 classifiers reach a plateau in accuracy, hence we used 60% of our data for training.

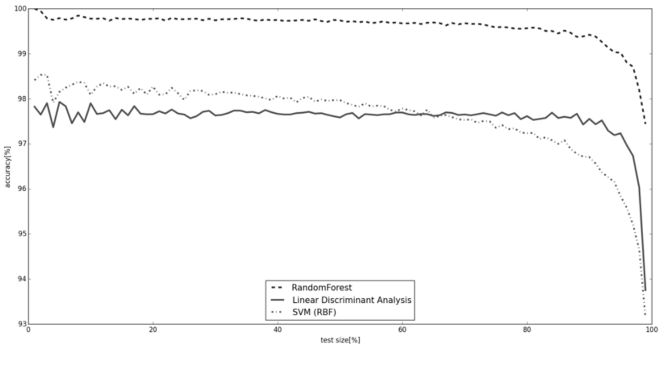


Fig. 2. Impact of the split of training and test data on the observed accuracy.

B. *Obfuscated vs. Non-Obfuscated*

To test the performance of the three machine learning classifiers, we conducted the following three experiments. For the first experiment, we used the **jsDelivr.com** data set and labeled the regular and minified files as non-obfuscated, and we labeled the files processed with one of the obfuscator tools as obfuscated. Figure 3a shows the classification results after 10-fold cross-validation and a split of 60%/40% for training and test datasets. On average, per cross-validation round, only 0.60% non-obfuscated scripts out of 3377 were classified as obfuscated and only 1.67% obfuscated scripts out of 3366 were classified as non-obfuscated.

In order to verify the performance of the classifiers trained with the jsDelivr.com dataset when deployed to classify a broad set of scripts from various sources, we used the JavaScripts found in the **Alexa Top 500** websites as the evaluation dataset. Figure 3b shows the results of this experiment. 99.4% of the scripts were labeled non-obfuscated and 0.6% as obfuscated. A manual inspection of the 52, 17 and 140 scripts for the SVM, random forest and LDA classifier revealed that about 30% of these scripts are true positives whilst 70% are false positives. The false negative (FN) rate is not measurable without manually classifying all the Alexa scripts. A manual inspection of a random sample of 50 of the scripts labeled as non-obfuscated did not contain obfuscated scripts. Based on this limited sample inspection we assume the database contains a negligible number of obfuscated scripts.

Figure 3 shows the results for the third experiment where we trained the classifiers with the full jsDelivr.com dataset and then tested them with the MELANI dataset. Since all of the scripts in the MELANI dataset are obfuscated and therefore have the correct label, not false positives are possible resulting in a precision of 100%. However, there are quite a lot of false negatives which results in low (for SVM) or even very low (for RF/LDA) recall.

C. *Malicious vs. Benign*

In the previous experiments, we focused on detecting obfuscation rather then maliciousness. Because our ultimate goal is to be able to distinguish between benign and malicious JavaScript, we trained and evaluated the classifiers to distinguish between benign and malicious scripts as well. The MELANI data set with malicious files is not large or representative enough for this result to be statistically significant. However, the results provide an indication whether the set of features and the machine learning approach can be used to detect malicious JavaScript. For this experiment, the jsDeliver.com and MELANI data sets were both serving as training and test data using a 10-fold cross-validation using a 60%/40% split. Figure 4b shows the results of this experiment. Less than one benign script has been classified as malicious per round and therefore precision and recall for benign samples is high.

		SVM	RF	LDA
Non Obfuscated	p	99.35%	99.81%	99.17%
	r	99.40%	99.97%	99.38%
	f1	98.87%	98.89%	99.27%
	s	33770	33770	33770
Obfuscated	p	99.40%	99.97%	99.37%
	r	98.33%	99.80%	99.16%
	f1	98.86%	98.89%	99.26%
	s	33660	33660	33660

(a) Classification results for the **jsDelivr.com** data set with 10-fold cross-validation and a split of 60%/40% for training and test data.

		SVM	RF	LDA
Non Obfuscated	p	100.00%	100.00%	100.00%
	r	99.45%	99.82%	98.52%
	f1	99.72%	98.91%	99.25%
	s	9459	9459	9459

(b) Classification results for the **Alexa Top 500** data set for the classifiers trained with the jsDelivr.com data set, based on the assumption that it contains no obfuscated scripts.

		SVM	RF	LDA
Obfuscated	p	100.00%	100.00%	100.00%
	r	60.61%	19.70%	22.73%
	f1	75.47%	32.91%	37.04%
	s	132	132	132

(c) Classification results for the **MELANI** data set when the classifiers are trained with the jsDelivr.com data set. All scripts in the MELANI data set are obfuscated but with different obfuscation techniques and tools than used in the jsDelivr.com data set.

Fig. 3. Performance of the classifiers with respect to distinguishing between obfuscated and non-obfuscated scripts, for different test data sets.

Figure 4 contains the results of our last experiment where we deployed a classifier trained with the jsDelivr.com and MELANI data sets to classify the Alexa Top 500 dataset. Since all of the scripts in this data set are labeled as benign, the precision is 100%. The recall of above 99% shows, that only a few of the Alexa Top 500 script were labeled as malicious. A verification of theses scripts did not reveal any malicious content.

V. DISCUSSION AND CONCLUSIONS

The results presented in Section IV-B give a summary of the evaluation of a machine-learning approach to distinguishing obfuscated vs. non-obfuscated JavaScripts. The results show that if the training dataset contains a representative set of samples of a particular type of obfuscation, it is likely to be reliably detected in the testing phase. One intriguing question presents itself: is the classifier learning a particular syntactic structure of a particular type of obfuscator, or do different types of obfuscation share some inherent characteristics that can be captured in the learning phase. The data in Figure 3b containing the results obtained for the Alexa 500 dataset suggest that the obfuscation techniques may share certain common syntactic characteristics. A more detailed analysis is required to identify the root cause and to improve classification

		SVM	RF	LDA
Benign	p	99.87%	99.84%	99.59%
	r	99.98%	100.00%	99.58%
	f1	99.93%	99.92%	99.59%
	s	67414	67414	67414
Malicious	p	97.65%	99.76%	48.63%
	r	83.88%	79.22%	49.08%
	f1	90.25%	88.31%	48.85%
	s	546	546	546

(a) Classification results for the combination of **jsDelivr.com** and the **MELANI** data set with 10-fold cross-validation and a split of 60%/40% for training and test data.

		SVM	RF	LDA
Benign	p	100.00%	100.00%	100.00%
	r	99.65%	99.97%	99.26%
	f1	99.83%	99.98%	99.63%
	s	9459	9459	9459

(b) Classification results for the **Alexa Top 500** dataset, when the classifiers are trained with the combined **jsDelivr.com** and **MELANI** datasets.

Fig. 4. Performance of the classifiers with respect to distinguishing malicious vs. benign scripts, for different test data sets.

results. However, low recall for the MELANI dataset 3 clearly suggests that there might be limits that could be challenging to overcome if custom obfuscation strategies or tools are used.

Our results presented in this paper suggest that it may be feasible to detect malicious JavaScripts. As the numbers reported in Figure 4 indicate, the precision and recall on the task of discriminating between malicious and benign scripts is high. These results, however, must be approached with caution. Since the database of malicious scripts was limited and much smaller than that of benign ones, it is not evident that the classifier is capturing the actual syntactic characteristics correlated with the malicious behavior. It is possible that the mere syntactic structure of the scripts in the MELANI database is sufficiently different from that of the jsDelivr.com to allow an accurate classification. As mentioned in the Introduction, the malicious script behavior is due to the script's functionality and not syntax per se. An in-depth analysis of the link between the functionality and syntax of a malicious code must be performed in order to deliver conclusive results.

We envision to continue our efforts towards understanding of the problem of automatic detection of malicious JavaScript code by collecting more representative datasets. Given such a dataset, an analysis of the statistical distribution of syntactic features and their dependence on the malicious code behavior will be studied. Consequently, we intent do develop a dedicated set of classification features insensitive to the type of obfuscation, which will allow for automatic detection of malicious JavaScript.

References

[1] W. Xu, F. Zhang, and S. Zhu, "The power of obfuscation techniques in malicious javascript code: A measurement study," in Malicious and Unwanted Software (MALWARE), 2012 7th International Conference on. IEEE, 2012, pp. 9–16.

[2] sven_t, "JSDetox," last accessed on 2016-01-30. [Online]. Available: http://www.relentless-coding.com/projects/jsdetox

[3] "Wepawet," last accessed on 2016-01-30. [Online]. Available: http://wepawet.iseclab.org/

[4] P. Likarish, E. Jung, and I. Jo, "Obfuscated malicious javascript detection using classification techniques," in Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on, Oct 2009, pp. 47–54.

[5] S. Kaplan, B. Livshits, B. Zorn, C. Siefert, and C. Curtsinger, ""nofus: Automatically detecting" + string.fromcharcode (32)+" obfuscated".tolowercase()+" javascript code," Microsoft Research, 2011.

[6] J. G. Politz, S. A. Eliopoulos, A. Guha, and S. Krishnamurthi, "Adsafety: Type-based verification of javascript sandboxing," CoRR, vol. abs/1506.07813, 2015. [Online]. Available: http://arxiv.org/abs/1506.07813

[7] W.-H. Wang, Y.-J. LV, H.-B. Chen, and Z.-L. Fang, "A static malicious javascript detection using svm," in Proceedings of the International Conference on Computer Science and Electronics Engineering, vol. 40, 2013, pp. 21–30.

[8] J. Lecomte, "Introducing the YUI Compressor," last accessed on 2016-01-30. [Online]. Available: http://www.julienlecomte.net/blog/2007/08/13/introducing-the-yui-compressor/

[9] M. Bazon, "UglifyJS," last accessed on 2016-01-30. [Online]. Available: http://lisperator.net/uglifyjs/

[10] D. Edwards, "Dean Edwards Packer," last accessed on 2016-01-30. [Online]. Available: http://dean.edwards.name/packer/

[11] "JavaScript Obfuscator," last accessed on 2016-01-30. [Online]. Available: http://javascriptobfuscator.com/

[12] P. Likarish and E. Jung, "A targeted web crawling for building malicious javascript collection," in Proceedings of the ACM first international workshop on Data-intensive software management and mining. ACM, 2009, pp. 23–26.

[13] "Alexa Top 500 Global Sites," last accessed on 2016-01-30. [Online]. Available: http://www.alexa.com/topsites

[14] L. Richardson, "Beautiful Soup," last accessed on 2016-01-30. [Online]. Available: http://www.crummy.com/software/BeautifulSoup/

[15] "DoubleKiller," last accessed on 2016-01-30. [Online]. Available: http://www.bigbangenterprises.de/en/doublekiller/

[16] B.-I. Kim, C.-T. Im, and H.-C. Jung, "Suspicious malicious web site detection with strength analysis of a javascript obfuscation," International Journal of Advanced Science and Technology, vol. 26, 2011, pp. 19–32.

[17] A. Hidayat, "Esprima JavaScript Parser," last accessed on 2016-01-30. [Online]. Available: http://esprima.org/

[18] "SpiderMonkey Parser API," last accessed on 2016-01-30. [Online]. Available: https://developer.mozilla.org/en-US/docs/Mozilla/Projects/SpiderMonkey/Parser_API

[19] R. O. Duda, P. E. Hart, and D. G. Stork, Pattern Classification, 2nd Edition, 2001.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, 2011, pp. 2825–2830.

Performance Study of a Software Defined Network Emulator

Jose M. Jimenez, Oscar Romero, Albert Rego, Avinash Dilendra, Jaime Lloret
Universidad Politécnica de Valencia
Camino Vera s/n 46022, Valencia (Spain)
email: jojiher@dcom.upv.es, oromero@dcom.upv.es, alrema91@gmail.com, avinash.dilendra@gmail.com, jlloret@dcom.upv.es

***Abstract*—Generally, network researchers use applications that allow them to emulate or simulate networks. It is desired to obtain very close results between the ones given in a virtual and the ones obtained when the real network hardware is implemented. In this paper, we compare the experimental results obtained when a virtual network is generated by using Mininet versus a real implemented network. We have compared them varying the Maximum Transmission Unit (MTU) on Internet Protocol version 4 (IPv4) and Internet Protocol version 6 (IPv6) packets. Ethernet, Fiber Distributed Data Interface (FDDI), and Wireless Local Area Network 802.11 (WLAN 802.11) MTUs have been used in our experimental tests.**

Keywords- SDN; OpenFlow; Mininet; MTU; virtualization; bandwidth; jitter.

I. INTRODUCTION

In the field of computer networks, the researches usually use programs that allow us to emulate or simulate networks. This is because, in most cases, we do not have the necessary devices needed to create complex networks. There are emulators and simulators as Omnet++ [1], OPNET [2], NS-2 [3], NS-3 [4] Netsim [5], GNS3 [6], etc. that are frequently used to create computer networks.

Deployment of network is very quick in virtual environment, even if it is needed a large number of resources, which is always practically almost impossible to implement with real hardware. Problem solving or troubleshooting capability is still easier than real implementations. Note that a network researcher has to keep in mind that the results obtained from a virtual network should be similar from those obtained by the real hardware network. If there is a significant difference between results of virtual network and real network, then the research work should not be taken into consideration. As a network test bed gives almost the same results than the real implemented network, then it saves a large amount of time, complexity and a lot of resources.

In general, network devices perform the transport and the control function. But, configuring a great amount of devices and changing the configuration efficiently to work properly, it means a big challenge for networking professionals.

Today's, computer network world is able to offer a large amount of functionalities suited to the requirements of users. A new technology, named Software Defined Networking (SDN) [7] appears to increase the efficiency and reduce the cost of network configuration.

Figure 1 shows the components of SDN in a layered structure. The first layer consists of some frequently used tools of monitoring and depuration. The tool “Oftrace” is used for analyzing and parsing Openflow message from network dump. “Oftrace” provides a library which analyzes and parses the message from TCP dump or Wireshark [8]. Loops or cyclic path can cause critical problems in SDN. “Oflops” is a tool to catch the loop mechanism in the software defined networks. It mentions the data packets in the loop which are not able to leave the network [9]. “Openseer” is a CGI script which helps to plot that data effectively in SDN [10]. In Controllers Layer there are few controllers which are used in SDN. More often, controllers are called the Brain of Network which controls and manages the software defined network. Floodlight, Open Daylight, Beacon, Nox are among the frequently used controllers in SDN [11]. Flow Visor ensures that multiple isolated logical networks can share the same topology and hardware resources of a network. It places as a transparent proxy between OpenFlow switches and OpenFlow controllers. The isolated logical network is named slice of the network and flow visor is named slicing software in SDN [12]. In SDN environment, OpenFlow switches are used to forward the packets. OpenFlow switches are either a software program or a hardware device which is compatible to OpenFlow protocols. Some of the commercial switches are available in market like HP, Nec, Juniper, etc. [13]. Mininet is used to create realistic virtual network within seconds on a single machine that could be able to run real kernel, switch and application code [14].

There are few emulators and simulators which are frequently used to run and control the technology SDN from a single screen, like NS-3, Estinet 9.0 [15], OmNet ++, Mininet, etc.

In this paper, we show the comparison among the obtained results from the virtual networks and from the real implemented networks. With the assessment of these results we are able to find the significant differences, which may be very useful for the researchers who all are performing their research work in Networking Industry.

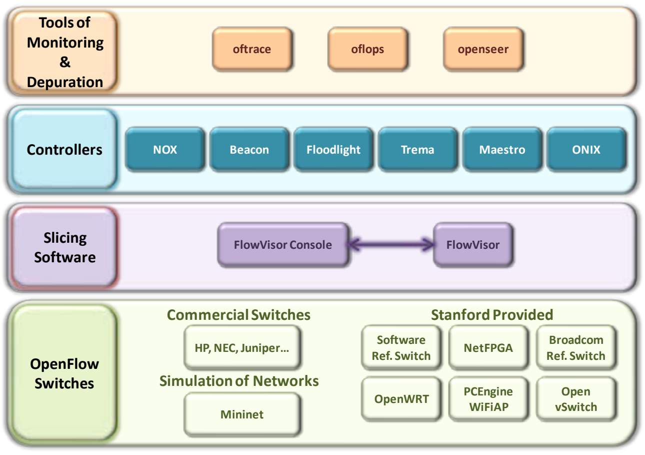


Figure 1.Key component of SDN in layered structure.

We have performed different experiments over Mininet and real implementation to have a good understanding of the network behavior in both scenarios. To do a detailed study, we must send data packets of different properties and compare the results.

We used the data packets with different Maximum Transmission Unit (MTU) on IPv4 and IPv6. These sizes of packets are usual for Ethernet version 2, Ethernet with LLC, PPoE, WLAN, Token Ring and FDDI.

The rest of the paper is structured as follows. In Section 2, we discuss existing related works. In Section 3, we introduced all resources that we used in our test bench. Measurement results and our discussion and analysis are shown in Section 4. Section 5 shows the conclusion and future works.

II. RELATED WORKS

In the past, a few researchers have accomplished their work in the area of SDN and investigated the performance of multimedia delivery over SDN. Furthermore, in the last years, emulators have been developed in order to provide an easy way to manage virtual networks and perform the research experiments. These emulators reduce the costs associated to the hardware needed to build the network. Inside the SDN research, the emulators have a great importance because of the great number of tests and the specific hardware that are necessary.

In the following section, we are going to discuss about some previous research work that helps us to get a deep understanding of SDN. Then, we will describe the previous researches in which emulators provide a useful way to test the experiments.

Recently, in our previous research article [16], we tried to evaluate the performance of multimedia streaming delivery over Mininet compared to real network implementation. We considered different properties of multimedia delivery, i.e., bandwidth, delay, jitter, and we found some significant differences over mininet and real test network. Kreutzet et al. [17] discussed the SDN, and analyzed the significance of SDN over traditional networking. Authors explained about the key components of SDN by using a bottom-up layered approach and focused on challenges, troubleshooting and debugging in SDN. Noghaniet et al. [18] introduced a framework based on SDN that could enable the network controller to deploy IP multicast between source and subscribers. The network controller was also able to control the distributed set of sources where multiple description coded (MDC) video content is available by using a simple northbound interface. Due to this SDN-based streaming multicast framework for medium and heavy workload, the Peak Signal-to-Noise Ratio (PSNR) of the received video is increasing considerably. Authors noticed that the received video, which had a very poor quality before, was having a significant increase in the quality of video now. Nam et al. [19] proposed a mechanism to solve the congestion problem and improve the video quality of experience (QoE). Authors tried to develop an SDN based application to improve the quality of video that can monitor conditions of network in real time streaming, and change routing paths dynamically by multi-protocol label switching (MPLS).

Egilmezet et al. [20] gives a unique design of an Openflow controller for multimedia delivery over SDN with end to end Quality of Service (QoS) support. The authors tried to optimize routes of multimedia dynamically. After experiments over real test network, the authors found better results than HTTP based multi-bitrate adaptive streaming. They ensured that OpenQoS can guarantee the video delivery with little or no video artifacts experienced by the end-users. In another publication, Egilmezet et al. [21] gave new distributed control plane architectures for multimedia delivery over large-scale, multi-operator SDN. The extension included in the design of architecture was: (a) to acquire network topology and the state information by topology aggregation and link summarization, (b) to propose an optimized framework for flowing based end to end over multi-domain networks, and (c) two distributed control plane designs by addressing the messaging between controllers for scalable and secure routing between two domains. By applying these extensions on layered video streaming, authors obtained a better quality of received video, reduced cost and memory overhead. This architecture was effectively scalable for large networks. Kassleret et al. [22] tried to negotiate the service and parameter for network communication between end users, and assign multimedia delivery paths in network according to prefixed service configuration. The idea behind this system was to centralized multi-user optimization of path assignments, which provide the better quality of experience by considering network topology, link capacities, delay and account service utility. Due to optimization, the system was able to use Openflow to set up forwarding paths in network.

III. TEST BENCH

In this section, we are going to introduce the SDN emulator and the real network topology used in our test bench.

A. *Devices and equipement*

In this subsection, we explain the devices and equipment used to perform our study.

The real topology is composed by the following equipment:

- 1 Switch Cisco Catalyst WS-C3560-24PS-E [23] that runs an IOS C3560-IPSERVICESK9-M,Versión12.2 (53) SE2, release software (fc3). It has 24 Fast Ethernet and 2 Gigabit Ethernet interfaces and 16 Mbytes of flash memory;
- 1 Desktop PC that has an Intel Core Quad Q9400 CPU @2.66 Ghz processor, 6 Gb of RAM memory, 1 Network Interface Card (NIC) Intel 82579V Gigabit Ethernet and Windows 7 Professional - 64 bits operative system;
- • 1 Desktop PC that has an Intel Core i5-2400 CPU @3.10 Ghz, 4 Gb RAM memory, 1 NIC Intel 82579V Gigabit Ethernet and Windows 7 Enterprise - 64 bits as operating system.

To design and develop the virtualized topology we have used a laptop composed by an Intel i7-4500UCPU @ 2.70 Ghz processor, 16 Gb RAM memory, 1 10/100/1000 Mbit/s NIC, and Ubuntu 14.04 - 64 bits as operating system.

B. *Software used*

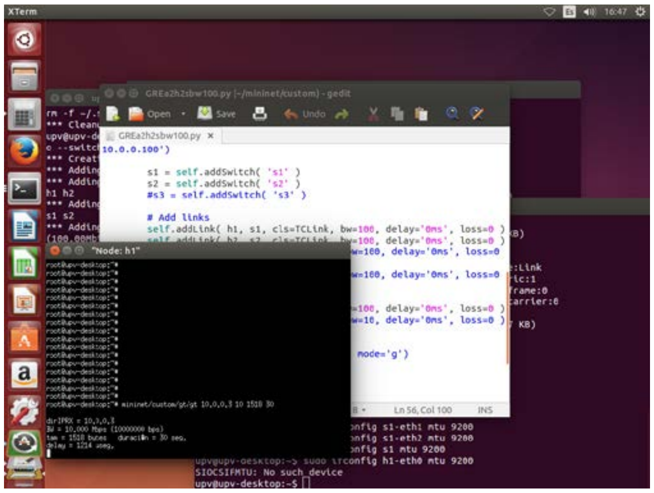
With Mininet, we can create a realistic virtual network, running real kernel, switch and application code, on a single machine. The machine can be a virtual machine, or a machine virtualized through the cloud, or a native machine. For our study, we have used Mininet version 2.2.1, with a native installation on Ubuntu 14 as shown in Figure 2.

We used a software application named gt, developed by us, to send traffic with different MTU and bandwidths.

In both, real and virtualized topologies, to capture and analyze the received traffic, we have used Wireshark [24], version 1.10.

C. *Characteristics of traffic transmited*

In our work, we send traffic with different MTUs that represents the packet sizes in different standards. Table I shows different sizes of MTU that was sent in our network topologies.



```
10.0.0.100')

        s1 = self.addSwitch( 's1' )
        s2 = self.addSwitch( 's2' )
        #s3 = self.addSwitch( 's3' )

        # Add links
        self.addLink( h1, s1, cls=TCLink, bw=100, delay='0ms', loss=0 )
        self.addLink( h2, s2, cls=TCLink, bw=100, delay='0ms', loss=0 )
```

```
root@upv-desktop:~# mininet/custom/gt/gt 10.0.0.3 10 1518 30

dirIPRX = 10.0.0.3
BW = 10.000 Mbps (10000000 bps)
tam = 1518 bytes   duración = 30 seg.
delay = 1214 useg.
```

```
rm -f ~/.
*** Clean
upv@upv-de
o --switc
*** Creat
*** Addin
*** Addin
h1 h2
*** Addin
s1 s2
*** Addin
(100.00Mb
```

```
onfig s1-eth1 mtu 9200
onfig s1-eth2 mtu 9200
onfig s1 mtu 9200
upv@upv-desktop:~$ sudo ifconfig h1-eth0 mtu 9200
SIOCSIFMTU: No such device
upv@upv-desktop:~$
```

Figure 2. Host running in Mininet.

As can be observed in Table I, sizes of MTU that was sent in our topology do not have standard values. This is because of the need to establish a GRE tunnel in the real topology, to connect the two hosts that have been created in Mininet, thus changing the frame size. Traffic was transmitted through UDP protocol. To calculate the jitter (J), we use the expression presented in RFC 4689 (Terminology for Benchmarking Network-layer Traffic Control Mechanisms) [25]. Therefore, we use the formula (1), where S_i is the transmission timestamp from packet i, and R_i is the reception timestamp of arrival packet i. For two consecutive packets i and j.

$$J = |(R_j - S_j) - (R_i - S_i)| \tag{1}$$

D. *Physical topology*

The real topology consists of two computer connected by straight-through cable, using one switch, as shown in Figure 3. The data transfer rates used is 10 Mbps.

In the virtualized network, we used a computer with Mininet, where we set up the same topology as the real one.

IV. MEASURAMENT AND DISCUSSION

This section, shows the results obtained in both cases, when traffic is being delivered over the real network and in the virtual topology using Mininet. Here, we present measures of traffic, with bandwidth links at 10 Mbps, and sending traffic at 10 Mbps. The parameters observed are bandwidth and jitter of packets with three different MTUs: 1518, 4370 and 7999, corresponding to the size of packets for Ethernet, FDDI and WLAN 802.11 traffic.

1) MTU - 1518: We present the results related of bandwidth and jitter obtained from the real and virtual topologies.

In Figure 4, we can see the bandwidth consumption values of the real topology and the values obtained in the virtual topology.

TABLE I. MTU PACKETS IN TOPOLOGIES

Frame Differentiation	
Media	**MTU (bytes)**
Ethernet wit LLC and SNP, PPPoE	1518
FDDI	4370
WLAN 802.11, Ethernet Jumbo Frame	7999

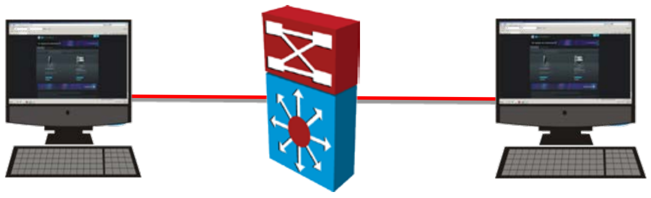


Figure 3. Real topology.

The data have similar values for both topologies when the transmission is stabilized. Although, in real topology is less than in virtual topology. The mean value of bandwidth in real topology is 9.5 Mbps while for virtual topology is 10 Mbps. The maximum and minimum values for real and virtual topologies are different, 9.9 Mbps and 20.8 Mbps for maximum and 6.7 Mbps and 9.9 Mbps for minimum. Observe that in the virtual topology, at the beginning of the transmission we obtain bandwidth values higher than 10 Mbps, meaning that in this situation the emulator is not accurate since the maximum bandwidth for a emulated 10 Mbps physical link should be 10 Mbps. After a few transmitted packets, the measured bandwidth is already providing more accurate values.

In Figure 5, we can see the jitter values of the real topology and the values obtained in the virtual topology. The values of the real topology are higher than those from the virtual topology. The mean value of jitter in real topology is 0.690 ms while for virtual topology is 0.001 ms. The maximum values real and virtual topologies are different, 3.169 ms and 0.607 ms. The minimum values for both topologies are the same, 0 ms.

2) MTU - 4370: We present the results related of bandwidth and jitter obtained for the real and virtual topologies.

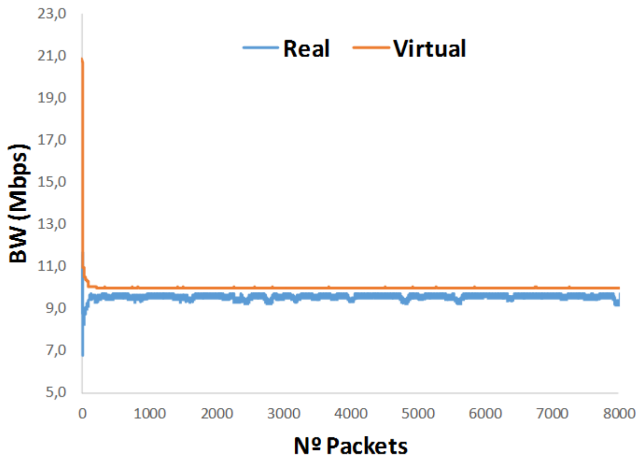


Figure 4. BW at 1518.

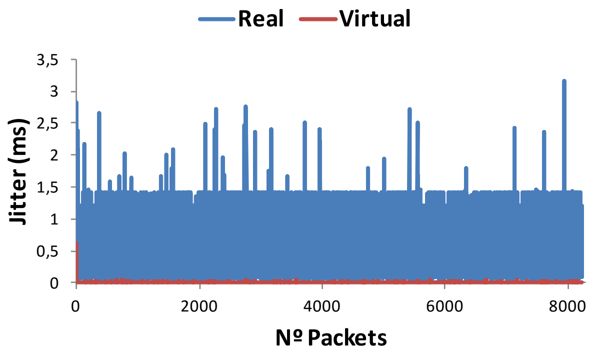


Figure 5. Jitter at 1518.

In Figure 6, we can see the bandwidth consumption values of the real topology and the values obtained in the virtual topology. The data have similar values for both topologies, when the transmission is stabilized, although in real topology is less than in virtual topology. The mean value of bandwidth in real topology is 9.5 Mbps while for virtual topology is 10 Mbps. The maximum and minimum values for real and virtual topologies are different, 9.8 Mbps and 31.5 Mbps for maximum, and 4.2 Mbps and 9.9 Mbps for minimum. As in the previous case, MTU 1518 bytes, in the virtual topology, we can observe that the bandwidth values are not realistic at the beginning of the transmission. After several transmitted packets, the values obtained are already close to the real network values.

In Figure 7, we can see the jitter values of the real topology and the values obtained in the virtual topology. The values of the real topology are higher than those from the virtual topology. The mean value of jitter in real topology is 0.228ms while for virtual topology is 0.002 ms. The maximum values for real topology are different, 9.189 ms and 1.277 ms. The minimum values for real topology and virtual topology are the same, 0 ms.

3) MTU - 7999: We present the results related of bandwidth and jitter obtained for the real and virtual topologies.

In Figure 8, we can see the bandwidth consumption values of the real topology and the values obtained in the virtual topology. The data have similar values for both topologies, when the transmission is stabilized, although in real topology is less than in virtual topology. The mean value of bandwidth in real topology is 9.5 Mbps while for virtual topology is 10 Mbps. The maximum and minimum values for real topology and virtual topology are different, 9.9 Mbps and 23 Mbps for maximum and 3.9 Mbps and 10 Mbps for minimum. Once again, the virtual topology is not providing realistic bandwidth values at the beginning of the transmission and, after transmitting a few packets, the bandwidth values are quite similar to those from the real network.

In Figure 9, we can see the jitter values of the real topology and the values obtained in the virtual topology. The values of the real topology are higher than those from the virtual topology. The mean value of jitter in real topology is 0.345 ms while for virtual topology is 0.001 ms. The maximum and minimum values for real topology and virtual topology are different, 25.091 ms and 0.844 ms for maximum and 0.037 ms and 0 ms for minimum.

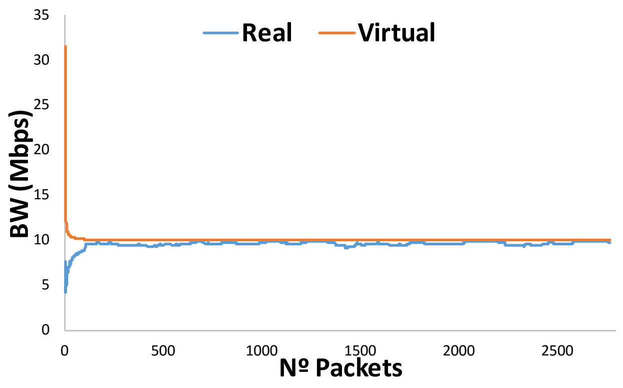


Figure 6. BW at 4370.

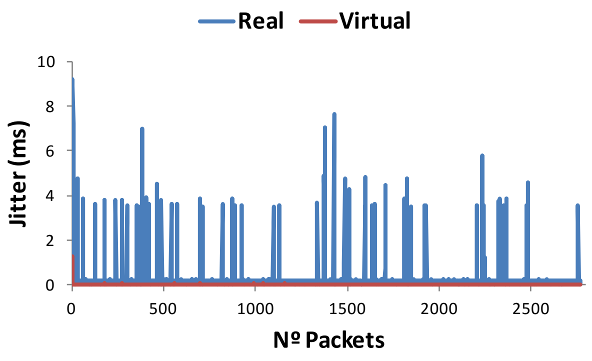


Figure 7. Jitter at 4370.

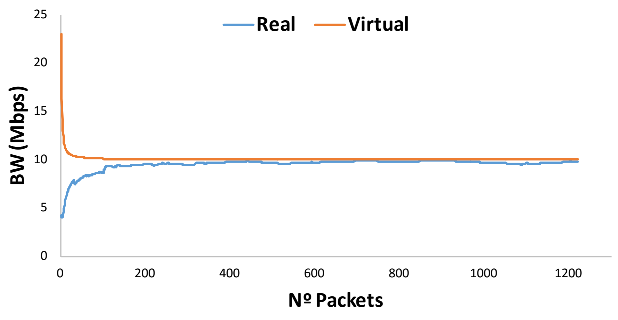


Figure 8. BW at 7999.

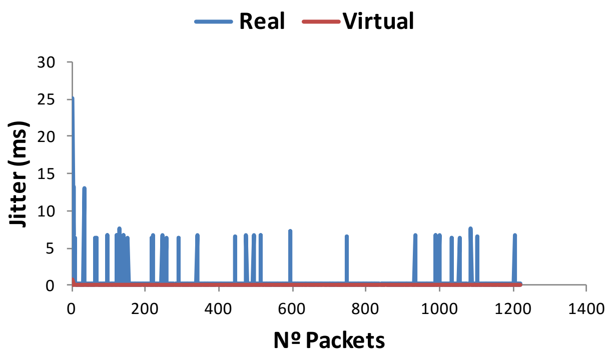


Figure 9. Jitter at 7999

V. Conclusion

In this paper, we have studied the transmission of different MTU sizes which correspond to Ethernet, FDDI, and WLAN 802.11 (also Jumbo Ethernet frames) packets. It can be seen that the variation of the bandwidth between the real and virtual topologies are very low. But, the results obtained for the jitter show that there are major deviations, although, we are working with a very low time scale, as we are dealing with milliseconds.

In our future work, we will compare real and virtual networks using more complex topologies, and by using Openflow compatible equipment.

.

Acknowledgment

This work has been supported by the “Ministerio de Economía y Competitividad”, through the “Convocatoria 2014. Proyectos I+D - Programa Estatal de Investigación Científica y Técnica de Excelencia” in the “Subprograma Estatal de Generación de Conocimiento”, Project TIN2014-57991-C3-1-P and the “Programa para la Formación de Personal Investigador – (FPI-2015-S2-884)” by the “Universitat Politècnica de València”.

References

[1] Omnet++. Available at https://omnetpp.org/ [Last access January 14, 2016]

[2] OPNET is now part of Riberved. Available at http://es.riverbed.com/products/performance-management-control/opnet.html / [Last access January 14, 2016]

[3] The Network Simulator - ns-2. Available at http://www.isi.edu/nsnam/ns/ [Last access January 14, 2016]

[4] NS-3. Available at NS-3 website: https://www.nsnam.org/ [Last access January 14, 2016]

[5] NetSim NETWORK SIMULATOR. Available at http://www.boson.com/netsim-cisco-network-simulator [Last access January 14, 2016].

[6] GNS3 The software that empowers networks professionals. Available at https://www.gns3.com/ [Last access January 14, 2016].

[7] Software-Defined Networking: A Perspective from within a Service Provider Environment. Available at: https:// tools.ietf.org/pdf/rfc7149.pdf [Last access January 14, 2016]

[8] Liboftrace. Available at http://archive.openflow.org/wk/index .php/Liboftrace [Last access January 14, 2016]

[9] OFLOPS. Available at https://www.sdxcentral.com/projects/ oflops/ [Last access January 14, 2016]

[10] OpenSeer. Available at http://archive.openflow.org/wk/ index.php/OpenSeer [Last access January 14, 2016]

[11] What are SDN Controllers (or SDN Controllers Platforms)?. Available at https://www.sdxcentral.com/resources/sdn/sdn-controllers/[Last access January 14, 2016]

[12] OpenFlow network virtualization with FlowVisor. Available at https://www.os3.nl/_media/2012-2013/courses/rp2/p28_ report.pdf [Last access January 14, 2016]

[13] OpenFlow switch. Available at http://searchsdn.techtarget. com/definition/OpenFlow-switch [Last access January 14, 2016]

[14] Mininet An Instant Virtual Network on your Laptop (or other PC). Available at http://mininet.org [Last access January 14, 2016]

[15] EstiNet Technologies Inc. Available at EstiNet Technologies website: http://www.estinet.com/index.php [Last access January 14, 2016]

[16] J. M. Jimenez, O. Romero, A. Rego, A. Dilendra and J. Lloret, “Study of Multimedia Delivery over Software Defined Networking” in Network Protocols and Algorithm, vol. 7, No. 4, 2015, pp. 37-62., 2015, doi:10.5296/npa.v7i48794

[17] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey", Proceedings of the IEEE, Volume103, Issue 1, Jan. 2015, pp. 14-76. 2015, http://dx.doi.org/10.1109/JPROC.2014.2371999

[18] K. A. Noghani and M. O. Sunay, “Streaming Multicast Video over Software-Defined Networks” Proceedings of the IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems (MASS) (2014), 28-30 Oct. 2014, pages 551-556, 2014, doi: 10.1109/MASS.2014.125

[19] H. Nam, K. Kim, J. Y. Kim and H. Schulzrinney, “Towards QoE-aware Video Streaming using SDN” Global Communications Conference (GLOBECOM), Dec. 2014, pp 1317-1322, 2014, doi: 10.1109/GLOCOM.2014.7036990

[20] H. E. Egilmez, S. T. Dane, K. Tolga Bagci and A. Murat Tekalp, “OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks” Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific, 3-6 Dec. 2012, Hollywood (USA), pp 1-8, 2012

[21] H. E. Egilmez, and A. M. Tekalp, ”Distributed QoS Architectures for Multimedia Streaming Over Software Defined Networks” Multimedia, IEEE Transactions on , Volume:16 , Issue: 6 , Sept 2014, pages: 1597 – 1609, 2014; doi:10.1109/TMM.2014.2325791

[22] A. Kassler, L. Skorin-Kapov, O. Dobrijevic, M. Matijasevic, and P. Dely, “Towards QoE-driven Multimedia Service Negotiation and Path Optimization with Software Defined Networking” Software, Telecommunications and Computer Networks (SoftCOM), IEEE, Sept 2012, Split (Croatia), pages: 1-5, 2012, ISBN: 978-1-4673-2710-7

[23] Z. Jingjing, C. Di, W. Weiming, J. Rong, and W. Xiaochun, “The Deployment of Routing Protocols in Distributed Control Plane of SDN” in The Scientific World Journal, Volume 2014, Article ID 918536, 8 pages, 2014, http://dx.doi.org/10.1155/2014/918536

[24] Wireshark software. Available at Wireshark website: https://www.wireshark.org/ [Last access January 14, 2016]

[25] Terminology for Benchmarking Network-layer Traffic Control Mechanisms. Available at https://www.ietf.org/rfc/rfc4689.txt.txt [Last access January 14, 2016]

A Study on How to Characterize TCP Congestion Control Algorithms from Unidirectional Packet Traces

Toshihiko Kato, Leelianou Yongxialee, Ryo Yamamoto, and Satoshi Ohzahata
Graduate School of Information Systems
University of Electro-Communications
Tokyo, Japan
e-mail: kato@is.uec.ac.jp, zoosiab@net.is.uec.ac.jp, ryo_yamamoto@is.uec.ac.jp, ohzahata@is.uec.ac.jp

Abstract— Although traffic in the Internet increases largely, it is sometimes pointed out that a small number of giant users exhaust large part of network bandwidth. In order to resolve such problems, a practical way is to suppress large traffic flows which do not conform to Transmission Control Protocol (TCP) congestion control algorithms. For this purpose, the network operators need to infer congestion control algorithms of individual TCP flows using packet traces collected passively in the middle of networks. We proposed, in our previous paper, a new scheme to characterize TCP algorithms from packet traces. It estimates the congestion window size (*cwnd*) at a TCP sender at round-trip time intervals, and specifies the *cwnd* growth as a function of the estimated value of *cwnd*. We showed that our previous scheme can characterize most TCP algorithms introduced recently. In an actual network environment, however, a packet trace captured over some link, especially a backbone link, often contains only unidirectional TCP segments due to the asymmetric routing. In this case, it is difficult to estimate the *cwnd* itself, and a new analysis scheme is required. This paper shows a study on how to characterize the TCP congestion control algorithms from unidirectional packet traces. We use a data size transmitted during a short period of time and, using it, we apply our former scheme to the unidirectional trace. This paper shows the results that we apply the proposed method to popular TCP algorithms, such as TCP Reno and CUBIC TCP.

Keywords- TCP congestion control; passive monitoring; unidirectional trace; congestion window.

I. INTRODUCTION

Recently, traffic in the Internet increases largely according to the increase of network capacity. The benefit of this capacity increase needs to be given equally to individual users. However, it is sometimes pointed out that some giant users exhaust large part of network bandwidth. Since most of traffic in the Internet uses TCP, the network congestions will be resolved by the TCP congestion control mechanisms. However, if any giant users do not conform to them intentionally, the problem will be worse. So, an important approach for network operators is to infer congestion control algorithms based on TCP segment exchanges captured over some link in the network. This is called the *passive approach* for TCP congestion control inferring. This is in contrast with the *active approach*, where an active tester sends test sequences to a target node and checks the replies

In the TCP congestion control [1], a data sender transmits data segments under the limitation of the congestion window size (*cwnd*) maintained within the sender itself, beside the advertised window reported from a data receiver. The value of *cwnd* grows up as a sender receives acknowledgment (ACK) segments and is decreased when it detects congestions. How to grow and decrease *cwnd* is the key of congestion control algorithm.

Although there were only a few congestion control algorithms, such as Tahoe, Reno and NewReno [2] in the early stage, many TCP congestion control algorithms have emerged recently [3]. For example, CUBIC TCP [4] and High Speed (HS) TCP [5] are designed for high speed and long delay networks. Among them, CUBIC TCP is used as a standard version in the Linux operating system. While many algorithms are based on packet losses, TCP Vegas [6] triggers congestion control against an increase of round-trip time (RTT). TCP Veno [7] combines loss based and delay based approaches such that the congestion control is triggered by packet losses but the delay determines how to grow *cwnd*.

This proliferation of algorithms complicates their inference in the passive approach. In our previous paper [8], we proposed a scheme for characterizing TCP congestion control algorithms from passively collected packet traces. As far as we know, this is the only passive approach which can handle most of the major TCP algorithms, differently from other proposals [9]-[14].

Our previous proposal requires both TCP data and acknowledgment segments are captured in a packet trace. It observes a RTT by mapping a data segment and its corresponding ACK segment, and estimates the value of *cwnd* as an outstanding data size during the RTT period. However, in an actual network environment, due to the asymmetric routing, a packet trace captured over some link, especially a backbone link, often contains only unidirectional TCP segments. In this case, it is difficult to estimate *cwnd*, and therefore, a new analysis scheme is required.

In this paper, we show a study on how to characterize the TCP congestion control algorithms from unidirectional packet traces. We propose an approach *based on sent data size during a short time period*. From the unidirectional trace, we cannot estimate either the RTT or *cwnd* values. Instead, we presume that the data size sent in a short time period is proportional to a *cwnd* value. Using this data size, we applied our former proposal to the unidirectional trace. We apply the proposed method to TCP Reno, CUBIC TCP, HS TCP, TCP Vegas, and TCP Veno.

The rest of this paper consists of the following sections. Section 2 surveys the related works. Section 3 discusses the detailed study on our proposal through applying the actual packet traces of TCP Reno and CUBIC TCP. Section 4 shows the results of our scheme being applied to other TCP versions. In the end, Section 5 gives the conclusions of this paper.

II. Related Works

The works on the passive approach TCP congestion control algorithm inference in the early stage [9][10] accepted an approach to keep track of the sender's *cwnd* based on the predefined TCP finite state machine. But, they considered only TCP Tahoe, Reno and New Reno and did not handle any of recently introduced algorithms. Oshio et al. [11] proposed a scheme to discriminate one out of two different TCP versions randomly selected from fourteen versions. They adopted an approach to keep track of changes of *cwnd* from a packet trace and to extract several characteristics, such as the ratio of *cwnd* being increased by one packet. But, they assumed that the discriminator knows which two TCP versions are used in the packet trace. Prior to our previous proposal [8], the only study which can infer the TCP algorithms including those introduced recently was a work by Yang et al. [15]. It is an example of the active approach. It makes a web server send 512 data segments under the controlled network environment, and observes the number of data segments contiguously transmitted. From those results, it estimates the window growth function and the decrease parameter to determine the TCP algorithm. All of the proposals so far, including our previous one, based on bidirectional TCP interactions.

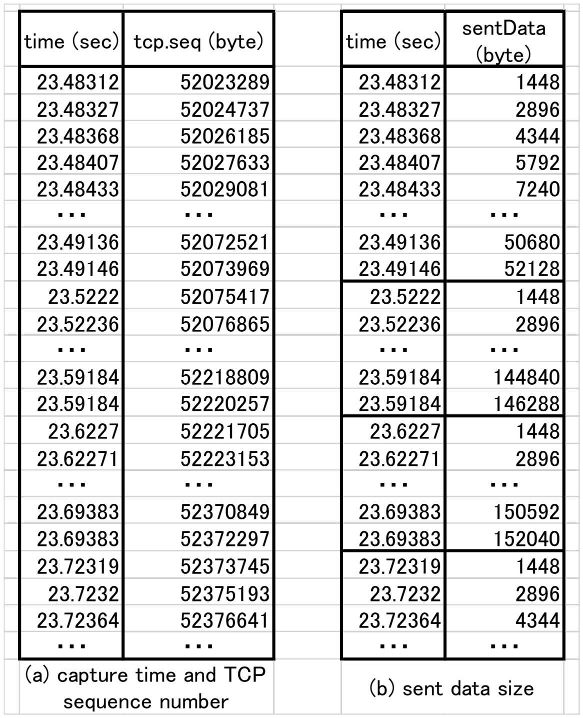
On the other hand, there are several works based on unidirectional packet traces [12]-[14]. T-RAT [12] used an approach to separate a unidirectional packet trace into flights, and then infer the TCP state of each flight (e.g., slow start or congestion avoidance). K. Lan and J. Heidemann [13] proposed an approach to examine characteristics of giant TCP flows in four dimensions, i.e., size, duration, rate, and burstiness, along with their correlations. Qian et al. [14] proposed a scheme to extract more detailed statistical features from unidirectional TCP traces. They focused on the size of initial congestion window, the relationship between the retransmission rate and the time required to transfer a fixed size of data for detecting the irregular retransmissions, and the flow clock to find TCP data transmissions controlled by the application or link layer factors. None of them, however, proposed a way to infer TCP algorithms from unidirectional packet traces.

In this paper, we discuss on a scheme to characterize recent TCP algorithms based on unidirectional traces. We use TCP Reno and CUBIC TCP as examples to design the scheme, and apply it to HS TCP, TCP Vegas and Veno.

III. Study by Analysing TCP Reno and CUBIC TCP

A. Proposal

In the rest of this paper, we use unidirectional packet traces which contain only the information on TCP data segments. From such a trace, we obtain a sequence of the time of individual packet capture and the TCP sequence number contained in captured packets for a specific TCP flow. Figure 1 (a) shows an example of such a sequence. This is selected from a CUBIC TCP trace.



time (sec)	tcp.seq (byte)
23.48312	52023289
23.48327	52024737
23.48368	52026185
23.48407	52027633
23.48433	52029081
・・・	・・・
23.49136	52072521
23.49146	52073969
23.5222	52075417
23.52236	52076865
・・・	・・・
23.59184	52218809
23.59184	52220257
23.6227	52221705
23.62271	52223153
・・・	・・・
23.69383	52370849
23.69383	52372297
23.72319	52373745
23.7232	52375193
23.72364	52376641
・・・	・・・

(a) capture time and TCP sequence number

time (sec)	sentData (byte)
23.48312	1448
23.48327	2896
23.48368	4344
23.48407	5792
23.48433	7240
・・・	・・・
23.49136	50680
23.49146	52128
23.5222	1448
23.52236	2896
・・・	・・・
23.59184	144840
23.59184	146288
23.6227	1448
23.62271	2896
・・・	・・・
23.69383	150592
23.69383	152040
23.72319	1448
23.7232	2896
23.72364	4344
・・・	・・・

(b) sent data size

Figure 1. Example of a unidirectional packet trace.

From this information, we use the following procedures to characterize the TCP algorithms.

- Check the sequence numbers and select retransmissions which can be detected by its decreasing.
- Pick up a portion where the sequence numbers are increasing continuously (no retransmissions occur). Figure 1 (a) is such a portion in a CUBIC TCP trace.
- Select a short time period to analyze the data size sent during this period. In Figure 1 (b), we select 100 msec as the period and apply it to the no retransmission portion given in (a) in this figure. The data size sent in a TCP segment is calculated by the TCP sequence number of the next segment.
- Use the data size sent during a selected time period as *sentData*. We use this as a value proportional to *cwnd*. In the result of Figure 1 (b), 52,128, 146,288, 152,040 (bytes) are those values.
- Calculate the difference of adjacent *sentData*. We denote it by *⊿sentData*. We presume this value is proportional to the increase of *cwnd*. In Figure 1 (b), 94,100 and 5,812 (bytes) are the values.
- Plot *⊿sentData* versus *sentData* graph and consult the result with the relationship obtained by our previous proposal [8]. We suppose that the result is similar with that in [8]. In the case of TCP Reno, for the estimated value of *cwnd* in a RTT interval, its difference *⊿cwnd* has the relationship with *cwnd* such as $\varDelta cwnd = 0 \; or \; 1$ (in unit of packet) independently of the value of *cwnd*. For, CUBIC TCP, the following relationship is resumed for *⊿cwnd* and cwnd.

$$\varDelta cwnd = 3RTT \cdot \sqrt[3]{C}\left(\sqrt[3]{cwnd - cwnd_{max}}\right)^2 \qquad (1)$$

Here, C is a predefined constant and $cwnd_{max}$ is the value of *cwnd* just before the last loss detection.

B. Study on Scheme Using TCP Reno

Next, we discuss the effectiveness of the proposal above through some experiments using actual packet traces. We use the packet traces collected in our previous study [8]. In our previous experiment, iperf TCP data transfer is performed between sending and receiving terminals. The sending terminal runs the Linux operating system and one of supported TCP versions is selected in the experiment. Those terminals are connected via a bridge, which inserts 100 msec delay (50 msec in one way) and packet losses whose probability is 1.0×10^{-4}. The sending terminal and the bridge are connected by a 100 Mbps Ethernet link. The receiving terminal and the bridge are connected by an Ethernet link or an IEEE 802.11g WLAN. The TCP segments transmitted are monitored by tcpdump at the sending terminal. We used either result of an Ethernet link or a WLAN depending on individual algorithms.

The obtained packet traces contain bidirectional TCP segments. From them, we selected only segments from the sending terminal to the receiving terminal, and obtained unidirectional packet traces such as one given in Figure 1 (a).

First, we tried a TCP Reno trace. We picked up a portion at the trace from 22.5 sec to 35.9 sec containing 11,437 segments whose sequence numbers continue to grow up. We selected 100 msec as the short time period for calculating *sentData*. The orange line in Figure 2 shows the relationship between *sentData* and time in this case. As this figure shows, the *sentData* is increasing linearly along with time. This result seems to reflect the behavior of TCP Reno correctly. Figure 3 shows the relationship between *ΔsentData* and *sentData*. The result indicates that *ΔsentData* mainly takes 0 byte and or 1448 bytes, and that it takes -1448 bytes and 2896 bytes sometimes. This result seems to be similar with that in our previous paper using bidirectional packet traces.

But, it needs to be mentioned that the period of 100 msec is equal to the round-trip delay inserted by the bridge in the experiment. So, the result in Figure 3 becomes similar to that in the previous paper.

As another value of time period, we use 200 msec. Figure 2 blue line shows the relationship between sent data and time in this case. The result is linear, but the value is twice of that in the above case. Figure 4 shows the relationship between *ΔsentData* and *sentData* obtained from the result above. In this case, the result shows that *ΔsentData* mainly takes 1448 bytes and or 4344 bytes. These values correspond to the segment size or three times of it. This result comes from the fact that the period for *sentData* calculation differs from the RTT value. We can say that, for TCP Reno, *ΔsentData* keeps constant irrelevant with *sentData* and takes two values mainly.

Figure 5 shows the relationship between *Δcwnd* and *cwnd* obtained in our previous paper in the same portion in the trace. The result in Figure 3 is similar with that in Figure 5. So, our scheme in this paper seems to be able to characterize TCP Reno from unidirectional packet traces. This is because the proposed scheme uses a time period identical to the delay inserted in the experiment. In Figure 4, the time period is 200 msec which is twice of the inserted delay. Still, there is a strong similarity among the result in Figure 4 and that in Figure 5, and so our scheme is working well in the case that the time period for calculating *sentData* is 200 msec.

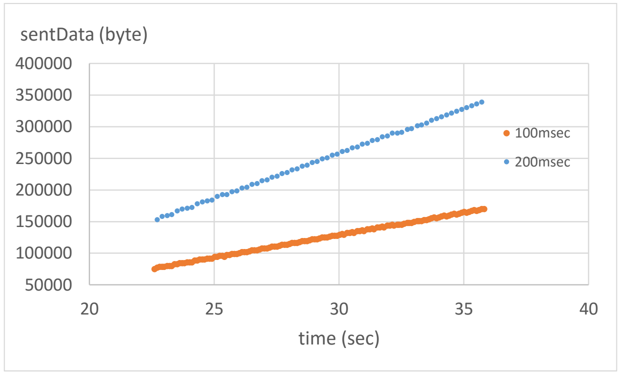


Figure 2. *sentData* vs. time for TCP Reno with 100 and 200 msec period.

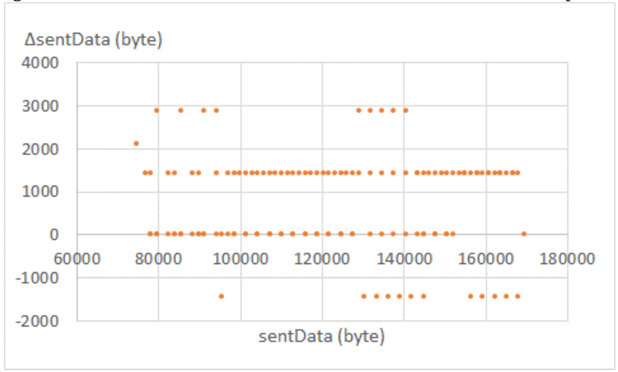


Figure 3. *ΔsentData* vs. *sentData* for TCP Reno with 100 msec period.

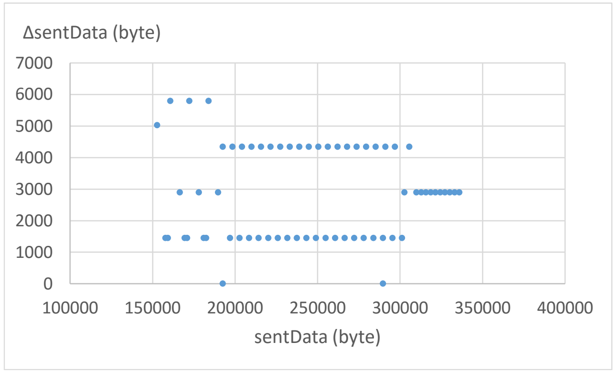


Figure 4. *ΔsentData* vs. *sentData* for TCP Reno with 200 msec period.

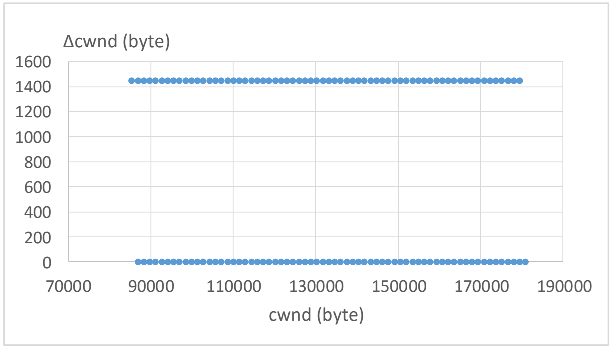


Figure 5. *Δcwnd* vs. *cwnd* for TCP Reno [8].

C. *Study on Scheme Using CUBIC TCP*

Next, we tried a CUBIC TCP trace as another example. As described above, our previous paper indicates that, in the case of CUBIC TCP, *⊿cwnd* has the relationship of the square of cubic root of *cwnd*. In order to examine whether this relationship can be applied to *⊿sentData* and *sentData*, we examined the unidirectional packet trace derived from the trace in our previous paper, with 100 msec and 200 msec time period for calculating *sentData*.

We picked up a portion at the trace from 23.5 sec to 38.3 sec containing 27,041 segments whose sequence numbers continue to grow up. Figure 6 shows the relationship between *sentData* and time for both cases of 100 msec and 200 msec time periods. In both cases, the *sentData* is increasing as a cubic curve of time.

Figures 7 and 8 show the relationship between *⊿sentData* and *sentData* for 100 msec and 200 msec time periods, respectively. Both figures indicate that *⊿sentData* is symmetrical for some value of *sentData* (200,000 and 400,000 bytes in Figures 7 and 8, respectively), and that *⊿sentData* is decreasing if *sentData* is smaller than the value and increasing if larger than the value. This situation is similar to the result of our previous result described in [9]. Especially, the situation is clearer for the case of 200 msec time period.

Figure 9 shows the relationship between *⊿cwnd* and *cwnd* obtained in our previous paper in the same portion in the trace. The results in Figures 7, 8 and 9 are similar and so we can say that the proposed scheme here works well for CUBIC TCP.

IV. Apply to Other TCP Versions

In this section, we apply our scheme to other TCP versions. In the following study, we use 200 msec as the time period for calculating *sentData* because we suppose that 200 msec will give smoother results.

A. *Result Applied to HS TCP*

HS TCP is designed to obtain high throughput over wide bandwidth and long delay networks. It grows *cwnd* to $cwnd + {}^{a(cwnd)}/_{cwnd}$ in response to every new ACK segment. The coefficient *a*(*cwnd*) is defined as 1, 2, and 3 when cwnd is 38, 118, and 221 (in unit of packet). So, in our previous approach, the estimated *⊿cwnd* will be as follows.

$$\triangle cwnd = \begin{cases} 0\ or\ 1\ (cwnd < 38) \\ 1\ or\ 2\ \ (38 \le cwnd < 118) \\ 1, 2\ or\ 3\ \ (118 \le cwnd < 221) \end{cases} \qquad (2)$$

Similarly with the study in Section 3, we made the unidirectional packet trace, and picked up a portion at the trace from 20.2 sec to 28.8 sec containing 2,749 data segments.

Figure 10 shows the relationship between *⊿cwnd* and *cwnd* obtained in our previous paper. Figure 11 shows the relationship between *⊿sentData* and *sentData*. These two figures are for the identical portion in the trace. From the results, it might be difficult to say that the method proposed in this paper can characterize the feature of HS TCP.

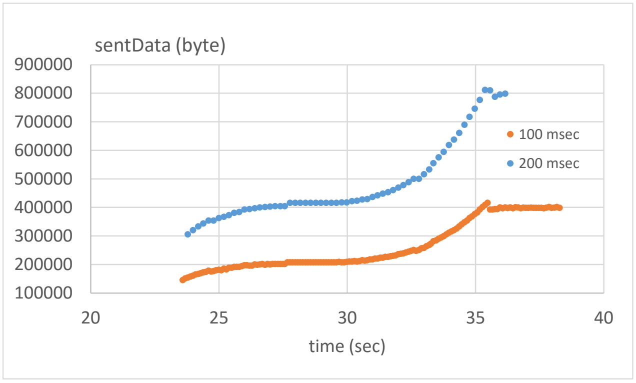


Figure 6. *sentData* vs. time for CUBIC TCP with 100 and 200 msec period.

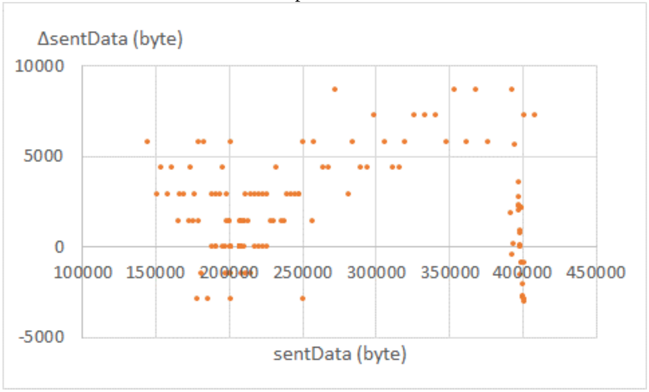


Figure 7. *⊿sentData* vs. *sentData* for CUBIC TCP with 100 msec period.

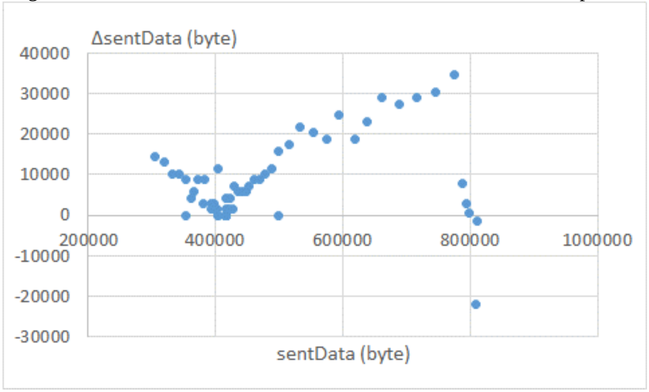


Figure 8. *⊿sentData* vs. *sentData* for CUBIC TCP with 200 msec period.

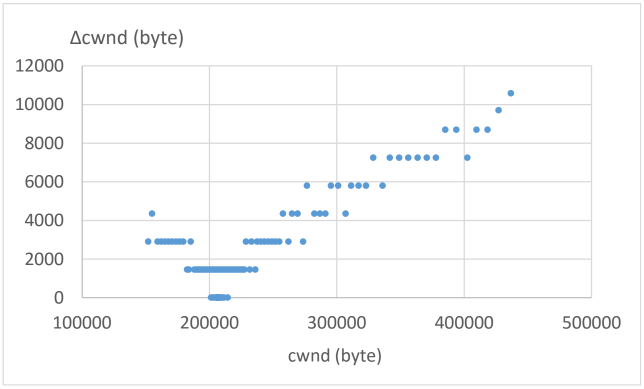


Figure 9. *⊿cwnd* vs. *cwnd* for CUBIC TCP [8].

B. *Result Applied to TCP Vegas*

TCP Vegas estimates the bottleneck buffer size using the current values of *cwnd* and RTT, and the minimal RTT for the TCP connection. At every RTT interval, Vegas uses this *BufferSize* to control *cwnd* in the congestion avoidance phase in the following way.

$$\varDelta cwnd = \begin{cases} 1 & (BufferSize < A) \\ 0 & (A \leqq BufferSize \leqq B) \\ -1 & (BufferSize > B) \end{cases} \quad (3)$$

Here, A = 2 and B = 4 (in unit of segment) are used in the Linux operating system (in unit of packet).

We made the unidirectional packet trace from the trace in the previous experiment, and picked up a portion at the trace from 38.0 sec to 59.7 sec containing 8,155 data segments.

Figure 12 shows the relationship between *Δcwnd* and *cwnd* obtained in our previous paper. Figure 13 shows the relationship between *ΔsentData* and *sentData*. For the values of *sentData* 100,000 through 120,000 bytes in Figure 13, *ΔsentData* are distributed between +10,000 bytes and –10,000 bytes. It can be said that there might be some similarities between the results of Figures 12 and 13 in those parts.

C. *Result Applied to TCP Veno*

TCP Veno (Vegas and ReNO) uses the *BufferSize* used by Vegas to adjust the growth of *cwnd* in the congestion avoidance phase as follows. If $BufferSize > B$ (*B* is the Vegas parameter *B*), cwnd grows by 1/cwnd for every other new ACK segment, and otherwise, it grows in the same manner with TCP Reno. Therefore, if the delayed ACK is not used, *Δcwnd* at RTT intervals will be as follows.

$$\varDelta cwnd = \begin{cases} 1 \ or \ 0 (BufferSize > B) \\ 1 \ (BufferSize \leqq B) \end{cases} \quad (4)$$

If the delayed ACK is used, $\varDelta cwnd = 0 \ or \ 1$ even if $BufferSize \leqq B$. But in this case, the ratio of *Δcwnd* being 1 and 0 is different for *BufferSize*. It will be 1:3 for *BufferSize*>*B*, and 1:1 for $BufferSize \leqq B$.

We made the unidirectional packet trace from the trace in the previous experiment, and picked up a portion at the trace from 37.6 sec to 52.7 sec containing 23,261 data segments.

Figure 14 shows the relationship between *Δcwnd* and *cwnd* obtained in our previous paper. Figure 15 shows the relationship between *ΔsentData* and *sentData*. From the results, it can be said that *ΔsentData* are flat independent of *sendData*. Although the values of *ΔsentData* are not fit to zero and one segment, it can be said that there might be some similarities between the results of Figures 14 and 15.

V. CONCLUSIONS

This paper presented some studies on how to characterize the TCP congestion control algorithms from passively collected unidirectional packet traces. We applied our previous scheme, which compares *cwnd*, estimated from bidirectional packet traces, and its difference. Since we cannot estimate *cwnd* from unidirectional traces, the proposed scheme in this paper selects a short time period and treats the sent data size during this period as being proportional to *cwnd* at this time. Our scheme characterizes the TCP algorithm by use of the graph of*ΔsentData* and *sentData*.

We applied this scheme to TCP Reno and CUBIC TCP in detail and showed that our proposal seems to work. We also applied our scheme to HS TCP, TCP Vegas and TCP Veno.

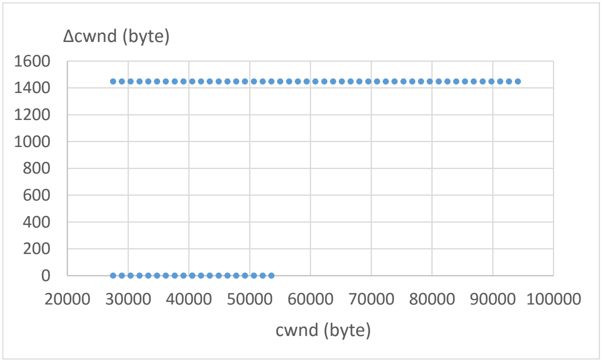


Figure 10. *Δcwnd* vs. *cwnd* for HS TCP [8].

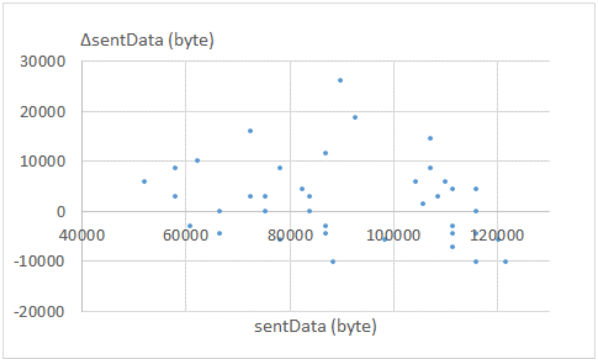


Figure 11. *ΔsentData* vs. *sentData* for HS TCP.

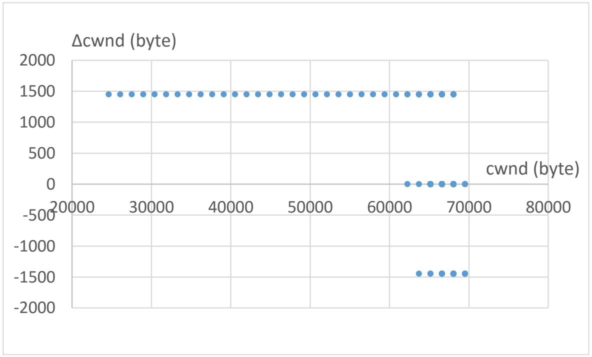


Figure 12. *Δcwnd* vs. *cwnd* for TCP Vegas [8].

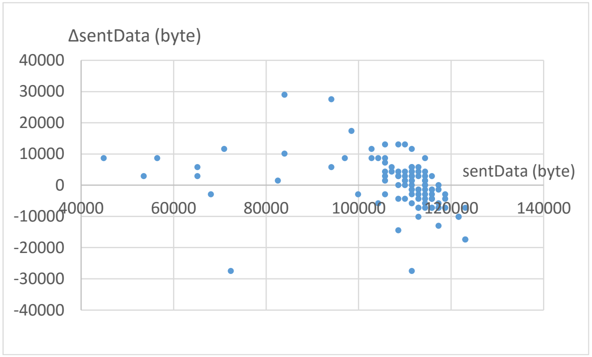


Figure 13. *ΔsentData* vs. *sentData* for TCP Vegas.

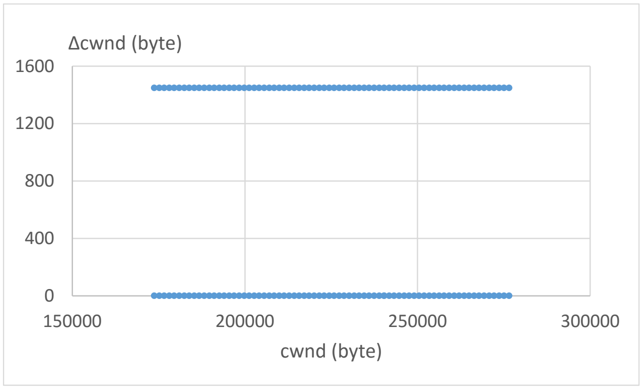


Figure 14. *Δcwnd* vs. *cwnd* for TCP Veno [8].

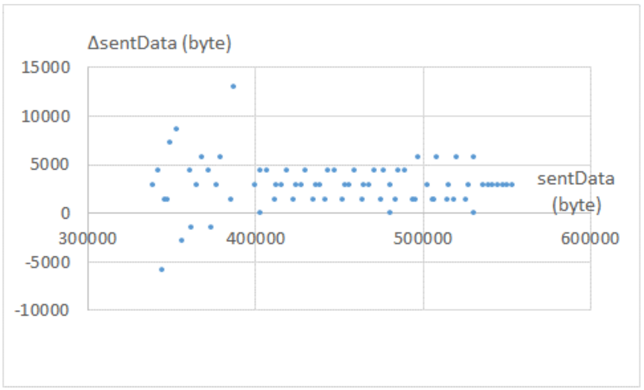


Figure 15. *ΔsentData* vs. *sentData* for TCP Veno.

The results were worse than those for Reno and CUBIC. For Vegas and Veno, the results in this paper have some similarity with the results in our previous paper, but there are some difficulties to distinguish Reno, Vegas and Veno. On the other hand, for HS TCP, our proposed scheme could not characterize it although our previous scheme could. It seems to characterize TCP Reno and CUBIC TCP algorithms well in our controlled in-laboratory setup. For the other TCP algorithms, the approach must be refined.

The points to be improved include the followings. First of all, the accuracy of our scheme needs to be increased. For this purpose, one important approach is to estimate a RTT correctly from unidirectional traces, because our scheme is sensitive for the fact that the short time period for calculating sent data is equal to RTT or integral multiples of RTT.

As describe above, the results in Figures 4, 13 and 15 have some similarities in the sense that the graphs have flat parts independent of *sentData*. So, it may be difficult to discriminate them. The next point is to invent a procedure to categorize the TCP algorithms into more general groups. , such as

- a flat type such as TCP Reno, HS TCP, TCP Westwood+ [16], TCP Vegas and Veno,
- a symmetric cubic root square type (CUBIC TCP),
- a monotonous increasing cubic root square type (Hamilton TCP [17]), and
- a random type (TCP Illinois [18]).

Thirdly, our previous scheme uses the estimation of multiplicative decrease parameter β is also used together with the estimation of window growth function. The scheme in this paper also needs to consider the multiplicative decrease parameter.

The last point is that the evaluation in this paper is done in a controlled laboratory setup. For the generalization of the results, the test has to be conducted under realistic network conditions

REFERENCES

[1] V. Javobson, “Congestion Avoidance and Control,” ACM SIGCOMM Comp. Commun. Review, vol. 18, no. 4, Aug. 1988, pp. 314-329.

[2] S. Floyd, T. Henderson, and A. Gurtov, “The NewReno Modification to TCP’s Fast Recovery Algorithm,” IETF RFC 3728, April 2004.

[3] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, “Host-to-Host Congestion Control for TCP,” IEEE Commun. Surveys & Tutorials, vol. 12, no. 3, 2010, pp. 304-342.

[4] S. Ha, I. Rhee, and L. Xu, “CUBIC: A New TCP-Friendly High-Speed TCP Variant,” ACM SIGOPS Operating Systems Review, vol. 42, no. 5, July 2008, pp. 64-74.

[5] S. Floyd, “HighSpeed TCP for Large Congestion Windows,” IETF RFC 3649, Dec. 2003.

[6] L. Brakmo and L. Perterson, “TCP Vegas: End to End Congestion Avoidance on a Global Internet,” IEEE J. Selected Areas in Commun., vol. 13, no. 8, Oct. 1995, pp. 1465-1480.

[7] C. Fu and S. Liew, “TCP Veno: TCP Enhancement for Transmission Over Wireless Access Networks,” IEEE J. Sel. Areas in Commun., vol. 21, no. 2, Feb. 2003, pp. 216-228.

[8] T. Kato, A. Oda, C. Wu, and S. Ohzahata, “Comparing TCP Congestion Control Algorithms Based on Passively Collected Packet Traces,” Proc. IARIA ICSNC 2015, Nov. 2015, pp.

[9] V. Paxson, “Automated Packet Trace Analysis of TCP Implementations,” ACM Comp. Commun. Review, vol. 27, no. 4, Oct. 1997, pp.167-179.

[10] S. Jaiswel, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, “Inferring TCP Connection Characteristics Through Passive Measurements,” Proc. INFOCOM 2004, March 2004, pp. 1582-1592.

[11] J. Oshio, S. Ata, and I. Oka, “Identification of Different TCP Versions Based on Cluster Analysis,” Proc. ICCCN 2009, Aug. 2009, pp. 1-6.

[12] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker, “On the Characteristics and Origins of Internet Flow Rates,” Proc. ACM SIGCOMM’02, Aug. 2002, pp. 309-322.

[13] K. Lan and J. Heidemann, “Measurement Study of Correlations of Internet Flow Characteristics,” Computer Networks, vol. 50, iss. 1, Jan. 2006, pp. 46-62.

[14] F, Qian, A. Gerber, and Z. Mao, “TCP Revisited: A Fresh Look at TCP in the Wild,” Proc. IMC ’09, Nov. 2009, pp. 76-89.

[15] P. Yang, W. Luo, L. Xu, J. Deogun, and Y. Lu, “TCP Congestion Avoidance Algorithm Identification,” Proc. ICDCS ’11, June 2011, pp. 310-321.

[16] L. Grieco and S. Mascolo, “Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control,” ACM Computer Communication Review, vol. 34, no. 2, April 2004, pp. 25-38.

[17] D. Leith and R. Shorten, “H-TCP: TCP for high-speed and long distance networks,” Proc. Int. Workshop on PFLDnet, Feb. 2004, pp. 1-16.

[18] S. Liu, T. Bassar, and R. Srikant, “TCP-Illinois: A loss and delay-based congestion control algorithm for high-speed networks,” Proc. VALUETOOLS ’06, Oct. 2006, pp. 1-13.

Intrusion Detection Using Indicators of Compromise Based on Best Practices and Windows Event Logs

María del Carmen Prudente Tixteco, Lidia Prudente Tixteco, Gabriel Sánchez Pérez, Linda Karina Toscano Medina
Instituto Politécnico Nacional
Sección de Estudios de Posgrado e Investigación ESIME Culhuacan
Santa Ana 1000, San Francisco Culhuacán, Coyoacán, D. F., México
mprudentet0900@alumno.ipn.mx, lprudente@ipn.mx, gasanchezp@ipn.mx, ltoscano@ipn.mx

***Abstract*— Nowadays computer attacks and intrusions have become more common affecting confidentiality, integrity or the availability of computer systems. They are more sophisticated making the job of the information security analysts more complicated, mainly because of the attacking vectors are more robust and complex to identify. One of the main resources that information security people have on their disposition are *Indicators of Compromise* (*IOCs*), which allow the identification of potentially malicious activity on a system or network. Usually IOCs are made off virus signatures, IP addresses, URLs or domains and some others elements, which are not sufficient to detect an intrusion or malicious activity on a computer system. The *Windows event logs* register different activities in a Windows® operating system that are valuable elements in a forensic analysis process. *IOCs* can be generated using *Windows event logs* for intrusion detection, improving *Incident Response* (*IR*) and forensic analysis processes. This paper presents a procedure to generate *IOCs* using *Windows event logs* to achieve a more efficient diagnostic computer system for *IR*.**

Keywords-indicators of compromise; windows event logs; intrusion detection.

I. Introduction

In the process of *IR* and forensic analysis, determining that a computer system is compromised could mean success or failure to mitigate a security incident. Attack vectors have increased their complexity, making more difficult for information security analysts to identify their presence. Different tools have been developed to facilitate their identification; one of these tools are the *IOCs*.

IOCs are pieces of forensic data, that could be found in log entries or system files, which can help to identify potentially malicious activity on a system or network [1].

In order to reduce the number of false-positive results over time, it is required to improve *IR* and forensic analysis procedures that collaborate directly with the *Computer Incident Response Team* (*CIRT*).

As a part of a forensic analysis recommendation of *Computer Emergency Response Teams* (*CERTs*), is of vital importance that a computer system intrusion is promptly identified, in order to perform the actions that will avoid further damages within the information system infrastructure [6].

Security Windows Event Logs Codes can be mainly used to describe malicious activity [4], and these can be added to antivirus signatures, IP addresses, URLs or domains to make *IOCs* more robust and specific to determine if a computer system could be compromised.

The *containment step* for the *IR* process is considered the most important; because it allows us to know if changes were made in the system [2], e.g., changes in privileges within registry keys, relocation of .dll system files or any other manipulation of processes and/or files during the intrusion on a computer system.

Knowing the behavior of the attacker is of great importance, it helps to achieve a better analysis stage for *IR* and forensic analysis processes, applying proactive actions and preventing future intrusions, e.g., computer system hardening.

Event logs should not be considered as isolated events; they have to be considered as a whole, that happen over a period of time and occur commonly, i.e., to obtain an *IOC* a failed login is not enough; this event must be associated with some others to determine a compromised computer system event.

This article presents a procedure to use *Windows event logs* to generate *IOCs*, achieving the detection of a computer system intrusion and help the information security people or a *CIRT* to take quickly and effective decisions to defend the information system infrastructure.

This paper is organized as follows. Section II presents state of the art. Section III presents an explanation of Windows event logs. Section IV describes the functionality of indicators of compromise. Section V describes the development of this research. Section VI presents the results of this research and Section VII conclusion and future work.

II. State of the Art

An incident is a compromise or a security violation in an organization. Preparation of the *CIRT* through planning, communication, and practice of the *IR* process will provide the experience needed to perform actions timely should an incident occur [2].

There are six basic steps when an *IR* occurs: preparation, identification, containment, eradication, recovery and lessons learned; they provide the basic foundation to create and perform their own *IR* plan. Specifically, there is one task within the containment phase that should be done: the system back-up. This provides the information about how the events happened, resulting in an incident from a malicious activity, or to be used for observing how the system was compromised during the phase of lessons learned [2].

In order for an *IR* to be considered successful, the *CIRT* should know the steps followed by an attacker when computer system is being compromised. There are seven steps: reconnaissance, weaponize, deliver, exploitation, installation, command and control (C2), actions on objectives, that the adversaries usually follow when attempting an intrusion, these leave a trail behind them [3]. This process is also known as the *Kill Chain Life Cycle* shown in Figure 1.

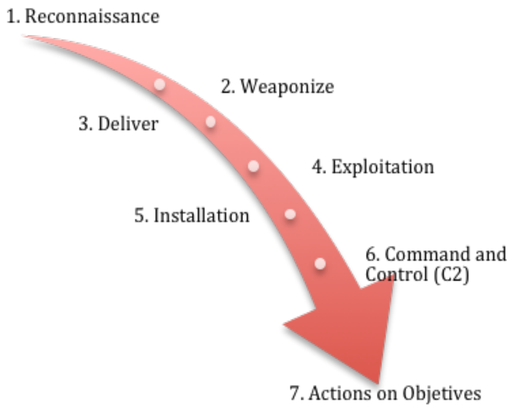


Figure 1. Kill Chain Life Cycle.

Microsoft Windows® Active Directory's best practices consider different signs to identify and evaluate a compromised computer system by monitoring and the use of alerts, through a proper configuration of Windows auditing settings. These signs can help to detect a malicious activity in a computer system early and timely.

The following security events can be considered as part of the event monitoring to detect possible signs of computer system intrusion within Windows® operating system [4]:

- Account Logon Events
- Account Management
- Directory Service Access
- Logon Events
- Object Access
- Policy Change
- Privilege Use
- Process Tracking
- System Events Audit

There are certain recommendations made by *CERTs* to help identify a compromised computer system. However, these have to be done by expert analysts [6]. When looking for signs of intrusion, most of the time if one host has been compromised, others on the same network have also been compromised. These are some of the signs to look for in a possible compromised computer system review [6]:

- Examine log files.
- Check for odd user accounts and groups.
- Check all groups for unexpected user membership.
- Look for unauthorized user rights.
- Check for unauthorized applications that start up automatically.
- Check for unauthorized processes.
- Check for altered permissions on files or registry keys.
- Check for changes in user or computer policies.
- Audit for intrusion detection.

On the other hand Hun-Ya Lock [5] presents the benefits of using OpenIOC framework as common syntax to describe the results of malware analysis; this work describes tools and techniques used during analysis but not in the reporting of results. The document emphasizes that reporting of the results is as important as the results themselves and if the results can be reported in a consistent well-structured manner that is easily understood by man and machine, then it becomes possible to automate some of the processes in the detection, prevention and reporting of malware infections.

IOC Editor is a free editor tool for *IOCs*. The *IOCs* are XML documents that support incident responders capturing different information about threats including malicious files attributes, changes in registries and artifacts in memory. IOC Editor provides an interface to manage data within these *IOCs* [7].

IOC Editor can:

- Manipulate the logical structures that define the *IOC*.
- Apply meta-information to *IOC* including detailed descriptions or arbitrary labels.
- Convert *IOCs* into XPath filters.
- Manage lists of terms that are used within *IOC*.

SANS [8] published a checklist to review critical logs during an *IR*. It can also be used for a log review routine.

The general approach is:

1) Identify which log sources and/or automated tools can be used during the analysis.

2) Copy log records over to a single location for later review.

3) Minimize noise by removing routinely and repetitive log entries after confirming that they are benign.

4) Determine whether logs' timestamps are reliable; considering the different time zone differences.

5) Focus on recent changes, failures, errors, status changes, access and administration events, and other unusual events in the environment.

6) Go back in time to reproduce the scenario before and after the incident.

7) Correlate activities across different logs to get a more comprehensive picture.

8) Develop theories about what occurred; explore logs to confirm or disprove them.

Some potential security log sources that can be found are [8]:

- Server and workstation operating system logs.

- Application logs.
- Security tool logs.
- Outbound proxy logs and end-user application logs.
- Other non-log sources for security events.

Finally, this document recommends what to look for on a Windows® OS like user logon/logoff events, user account changes, password changes, startup or stopping of a service, object access denial and other kind of resources [8].

Windows event logs can be used to generate *IOCs* and help identify should a computer system is compromised, detecting possible intrusions or strange behaviors, and make timely decisions accordantly.

None of the above references use *IOCs* in conjunction with *Windows event logs*.

This paper presents a procedure to fix this situation.

The efficiency of these *IOCs* could be improved if correlation tools are implemented to analyze attack vectors promptly.

III. Windows Event Logs

This following section describes key concepts about the Windows event logs.

A. Log

A log is a record of events occurring within an organization's systems and networks. Logs are composed of entries and they have evolved to contain information related to many different kinds of events [9].

B. Operating Systems Logs

Usually, Operating Systems (OS) for computers log a variety of information related to security. The most common types of security-related OS data are the following:

- System Events. System events are operational actions performed by OS components. Many OS allow administrators to specify what type of events will be logged and what kind of details register, such as timestamp, status and error codes, service name and user or system account associated with each event.
- Audit Records. Audit records contain security event information such as successful and/or failed authentication attempts, file accesses, security policy changes, account changes, and use of privileges [9].

OS logs are the most beneficial to identify or investigate suspicious activity involving a particular host. After suspicious activity like attacks, frauds, and inappropriate usage, OS' logs can be consulted to obtain more information on the activity and type of situation. Commonly they contain detailed information about each activity. Other logs can contain information less detailed and are helpful only to correlate events recorded in the primary log types [9].

C. Windows Event Log

Windows event log is a record of events that happen on a computer system, generating alerts and notifications. Microsoft® [10] defines an event as "any significant occurrence in the system or in a program that requires users to be notified, or an entry added to a log".

Some Windows event logs categories are [11]:

- Application. Events in this log are classified as error, warning, or information, depending on the severity of the event. An error is a significant problem. A warning is an event that is not necessarily significant, but might indicate a possible future problem. An information event describes the successful operation of a program, driver, or service.
- Security. This log contains security-related events, which are called audit events, and are described as successful or failed, depending on the event.
- Setup. Computers that are configured will have additional logs displayed here.
- System. System events are sent to this log by Windows and Windows system services, and are classified as error, warning, or information.
- Forwarded Events. Events are forwarded to this log by other computers.

Windows operating system classifies events by type as:

- Information event. Describes the successful completion of a task.
- Warning event. Notifies the administrator of a potential problem.
- Error message. Describes a significant problem that may result in a loss of functionality.
- Success audit event. Indicates the completion of an audited security event.
- Failure audit event. Describes an audited security event that did not complete successfully, such as an end-user locking himself out by entering incorrect passwords [10].

Each event in a log entry contains the following information:

- Date. The date the event occurred.
- Time. The time the event occurred.
- User. The user name of the user who was logged on when the event occurred.
- Computer. The name of the computer.
- Event ID. A Windows identification number that specifies the event type.
- Source. The program or component that caused the event.
- Type. The type of event [10].

For the purpose of this research different event logs of each category were selected and organized in the following tables. These events are the most representative of each of the categories being registered in a computer system and can help to generate an *IOC*. Tables I to VIII show the association of current Windows *Event ID* with its *Event Summary*, retrieved from appendix of Events to Monitor by Microsoft [13]:

TABLE I. EVENTS OF ACCOUNT LOGON CATEGORY.

Event ID	Event Summary
4774	An account was mapped for logon.
4776	The domain controller attempted to validate the credentials for an account.

TABLE II. EVENTS OF ACCOUNT MANAGEMENT CATEGORY.

Event ID	Event Summary
4783	A basic application group was created.
4785	A member was added to a basic application group.
4741	A computer account was created.
4742	A computer account was changed.
4727	A security-disabled global group was created.
4728	A member was added to a security-disabled global group.
4720	A user account was created.
4722	A user account was enabled.
4724	An attempt was made to reset an account's password.
4738	A user account was changed.
4740	A user account was locked out.

TABLE III. EVENTS OF PROCESS TRACKING CATEGORY.

Event ID	Event Summary
4688	A new process has been created.
4689	A process has exited.

TABLE IV. EVENTS OF LOGON CATEGORY.

Event ID	Event Summary
4634	An account was logged off.
4647	User initiated logoff.
4624	An account was successfully logged on.
4625	An account failed to log on.
4649	A replay attack was detected. May be a harmless false positive due to misconfiguration error.
4778	A session was reconnected to a Window Station.
4801	The workstation was unlocked.
4964	Special groups have been assigned to a new logon.

TABLE V. EVENTS OF OBJECT ACCESS CATEGORY.

Event ID	Event Summary
4665	An attempt was made to create an application client context.
4668	An application was initialized.
4664	An attempt was made to create a hard link.
4985	The state of a transaction has changed.
5051	A file was virtualized.
4691	Indirect access to an object was requested.
4698	A scheduled task was created.
4700	A scheduled task was enabled.
4702	A scheduled task was updated.
4657	A registry value was modified.
4660	An object was deleted.

TABLE VI. EVENTS OF POLICY CHANGE CATEGORY.

Event ID	Event Summary
4719	System audit policy was changed.
4905	An attempt was made to unregister a security event source.
4907	Auditing settings on object were changed.
4912	Per User Audit Policy was changed.
4704	A user right was assigned.
4946	A change has been made to Windows Firewall exception list. A rule was added.
4947	A change has been made to Windows Firewall exception list. A rule was modified.
4948	A change has been made to Windows Firewall exception list. A rule was deleted.
4949	Windows Firewall settings were restored to the default values.
4950	A Windows Firewall setting has changed.
4670	Permissions on an object were changed.

TABLE VII. EVENTS OF PRIVILEGE USE CATEGORY.

Event ID	Event Summary
4672	Special privileges assigned to new logon.
4673	A privileged service was called.

TABLE VIII. EVENTS OF SYSTEM AUDIT CATEGORY.

Event ID	Event Summary
5025	The Windows Firewall Service has been stopped.
5034	The Windows Firewall Driver has been stopped.
4697	Attempt to install a service
4618	A monitored security event pattern has occurred.

The events review from Windows logs can help trace the activities, *IR* and keep computer systems secure. Configuring these logs properly can help to manage the logs efficiently to identify and diagnose the current source of system problems and predict future ones.

IV. INDICATORS OF COMPROMISE

It is necessary to know the terminology used for indicators of compromise.

- Expression. The definition of a condition which, when true, suggests that intrusion activity is present.
- Simple Expression. An expression that can be defined without using "AND" or "OR" logic operators.
- Complex Expression. An expression that combines multiple simple expressions using "AND" or "OR" logic operators.

- Indicator of Compromise (*IOC*). A mix of expressions (simple, complex, or both), usually grouped together for the purpose of describing a single piece of malicious activity [7].

Figure 2 shows the common *IOC* structure.

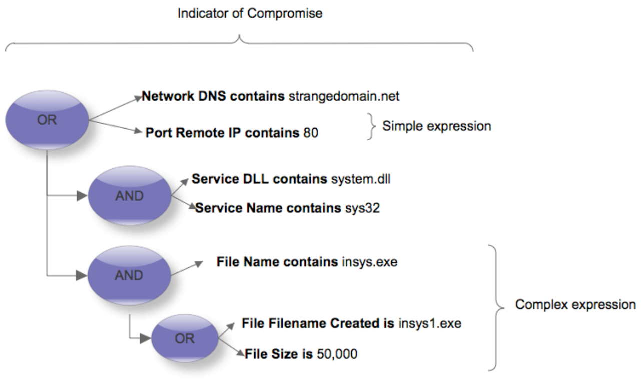
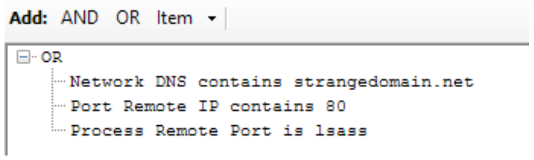

Indicator of Compromise
OR
Network DNS contains strangedomain.net
Port Remote IP contains 80
Simple expression
AND
Service DLL contains system.dll
Service Name contains sys32
AND
File Name contains insys.exe
OR
File Filename Created is insys1.exe
File Size is 50,000
Complex expression

Figure 2. IOC structure.

A. IOC Editor Logic

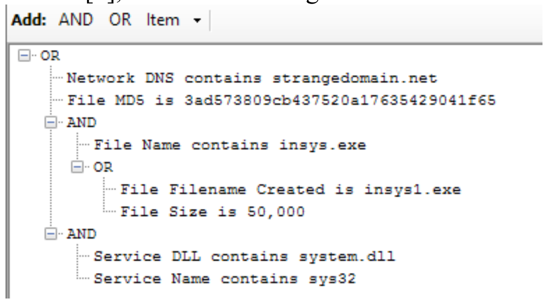
An *IOC* can be represented by expressions on a logic tree. The logic tree starts out with a top-level "OR" structure. When expressions are added, an *IOC* will be triggered as long as one of the expressions describes a true circumstance. Sometimes an *IOC* will consist of a collection of simple expressions listed in the top-level "OR" structure without the need of a more complex logic tree [7], as shown in Figure 3.



```
⊟ OR
   Network DNS contains strangedomain.net
   Port Remote IP contains 80
   Process Remote Port is lsass
```

Figure 3. IOC by Simple logic.

When required, logic branches can be built with "AND" and "OR" substructures to form complex expressions. Each "AND" and "OR" applies to the branches only in its substructure [7], as is shown in Figure 4.



```
⊟ OR
   Network DNS contains strangedomain.net
   File MD5 is 3ad573809cb437520a17635429041f65
   ⊟ AND
      File Name contains insys.exe
      ⊟ OR
         File Filename Created is insys1.exe
         File Size is 50,000
   ⊟ AND
      Service DLL contains system.dll
      Service Name contains sys32
```

Figure 4. IOC by Logic branch.

OpenIOC is an open source framework developed by Mandiant® for sharing threat intelligence [12]. It can be used to improve the reliability and repeatability of forensic analysis, to support the investigations of incidents or suspicious malicious activity in the IT operations of an organization.

V. DEVELOPMENT

This section describes the steps followed to determine some *IOCs* using *Windows event logs* that allow early detection of malicious activity.

Using the classification made by Microsoft [4], only eight out of nine of the categories were used (shown in Figure 5). These categories can be associated with general events, and used to detect malicious intrusion events on a computer system.

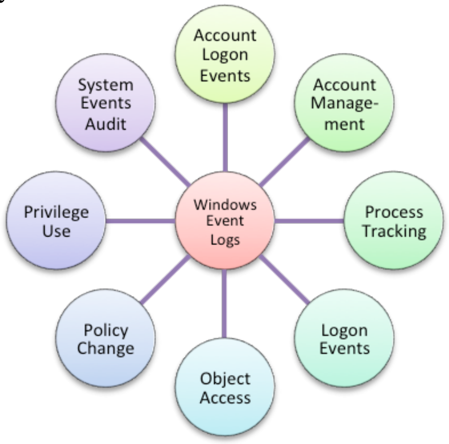


Figure 5. Categories used of *Windows event logs*.

In order to generate *IOCs* using *Windows event logs*, the main events (as described by Microsoft [4], and Anton Chuvakin et al. [8]) were reviewed, these are user-related, system access, granting of permissions or privileges, and activities or services modified in the system whilst an intrusion occurred.

The following sentences describe some of the possible actions that an external agent would perform upon an intrusion, and how this is reflected in the *Windows event logs*:

1) If an unauthorized user tries to access the system with invalid credentials, Windows event log provides the information regarding the number of attempts to access the computer system and the user credentials that are being used on a specific period of time. Examples of the information provided are: "*authentication failure*" or "*failed to log on*".

2) While the anomalous agent has valid credentials, successful login events will be also considered.

Then the anomalous agent, within the system and depending on the level of privileges the account has, will try:

3) Privileges Escalation, administrative privileges could be granted to ordinary accounts in order to create other user and/or system accounts; execute actions amongst user groups that could be reflected on events related to creation or modification of accounts and eventually taking total control over the system and be able to disable or delete existing accounts.

4) Change in *Windows Security Audit*, preventing modifications made within the system to be identified.

5) Create or modify files and/or system objects belonging to the installation process itself.

6) Disable or change *Windows Firewall Settings* to allow communications between malicious domains.

7) Change policies for network connections.

8) Install applications to create new services or change services already running in the system.

9) Delete audit events to prevent register activities in the system.

10) Log out session.

Having acquired all possible actions made by an intruder, we considered the odds of each event happening continuously over a determined period of time to classify them in the following activities: user, audit, services and objects (shown in Figure 6).

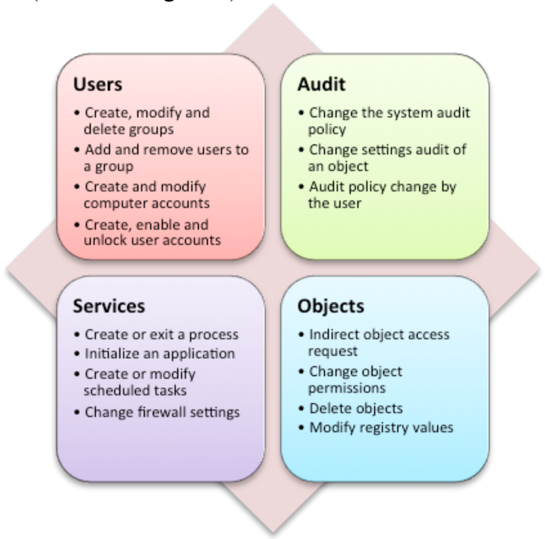

Users
• Create, modify and delete groups
• Add and remove users to a group
• Create and modify computer accounts
• Create, enable and unlock user accounts
Audit
• Change the system audit policy
• Change settings audit of an object
• Audit policy change by the user
Services
• Create or exit a process
• Initialize an application
• Create or modify scheduled tasks
• Change firewall settings
Objects
• Indirect object access request
• Change object permissions
• Delete objects
• Modify registry values

Figure 6. Classification of events by their activity.

Considering that events treated individually, might indicate a normal behavior on a computer system, it is necessary to generate a different kind of *IOC*, which could handle different events in conjunction that could determine malicious activity.

The following is a case study that represents the procedure to generate an *IOC* using *Windows event logs*:

"Peter" is a valid user (the User from now on) in a computer system with Windows® 8 OS. Peter's account has read-only user privileges.

After a security incident, the following activities were detected:

1) Two attempts to login as the User were executed.

2) User session started successfully.

3) Special privileges were assigned to User's account.

4) A new user account was created, named "Jame".

5) A global group with security-disabled settings was created.

6) An explorer process has been created.

7) An attempt to unregister a security event source was executed.

8) Jame's account was enabled.

9) The auditing settings on access-control object were changed.

10) Peter´s account session was closed.

The diagram in Figure 7 shows the activity sequence of an *IOC* considering the possible events achieved by an intruder using Peter's account.

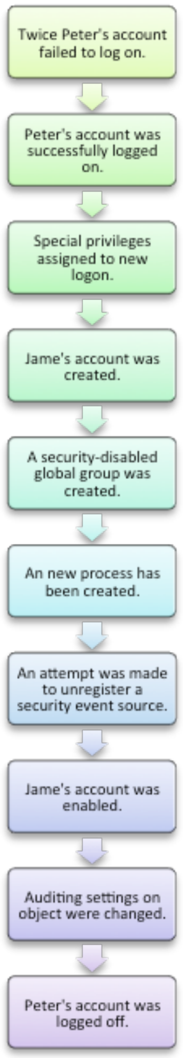

Twice Peter's account failed to log on.
Peter's account was successfully logged on.
Special privileges assigned to new logon.
Jame's account was created.
A security-disabled global group was created.
An new process has been created.
An attempt was made to unregister a security event source.
Jame's account was enabled.
Auditing settings on object were changed.
Peter's account was logged off.

Figure 7. Simple event sequence by malicious activity of a case study.

The next step on this procedure is to identify the event ID corresponding to the activities registered on the sequence considered malicious. Figure 8 shows the sequence of current Windows event ID for this case study.

Then an IOC could be generated to analyze the malicious activity on future occasions. It could be described with the following *simple expression* using Windows event ID:

AND (4625, 4625, 4624, 4672, 4720, 4728, 4688, 4905, 4722, 4907, 4634)

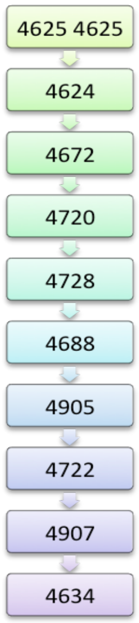
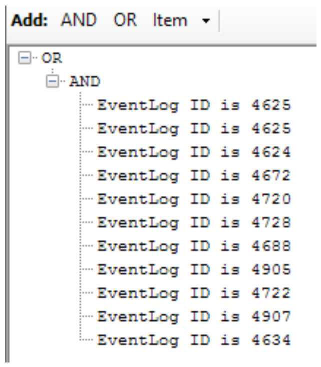


Figure 8. Event ID sequence by malicious activity of a case study.

Through the use of the IOC Editor tool, different indicators of compromise can be developed for the purpose of a timely identification of possible intrusions on Windows® OS. The *simple expression* for the case study developed using IOC Editor can be represented as shown in Figure 9.



```
OR
  AND
    EventLog ID is 4625
    EventLog ID is 4625
    EventLog ID is 4624
    EventLog ID is 4672
    EventLog ID is 4720
    EventLog ID is 4728
    EventLog ID is 4688
    EventLog ID is 4905
    EventLog ID is 4722
    EventLog ID is 4907
    EventLog ID is 4634
```

Figure 9. IOC of simple expression using IOC Editor.

Different *IOCs* could be generated using statements that represented different sequences of malicious activities previously registered. Figure 10 shows another example; taking into account the case study previously explained and considering other event sequences, a more efficiently *IOC* can be generated, reducing the false-positive behavior for a forensic analysis.

Considering the different options of events described for this new example, Figure 11 shows the current Windows event ID for current activities sequences.

To represent sequences identified in the latest example, simple expressions are combined using AND and OR logical operators, to generate an *IOC* of *complex expression*.

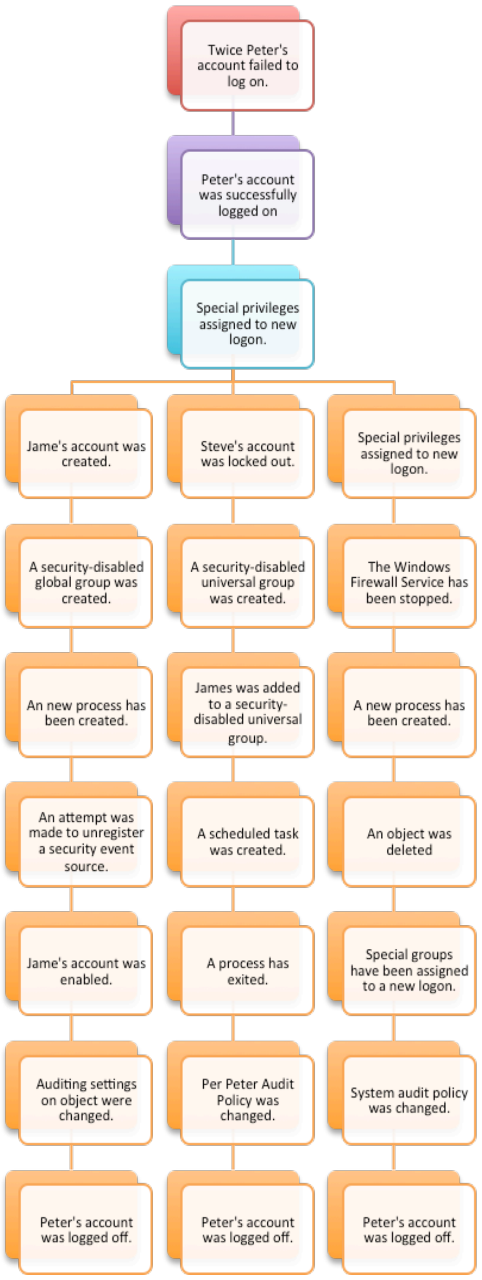

Twice Peter's account failed to log on.
Peter's account was successfully logged on
Special privileges assigned to new logon.
Jame's account was created.
Steve's account was locked out.
Special privileges assigned to new logon.
A security-disabled global group was created.
A security-disabled universal group was created.
The Windows Firewall Service has been stopped.
An new process has been created.
James was added to a security-disabled universal group.
A new process has been created.
An attempt was made to unregister a security event source.
A scheduled task was created.
An object was deleted
Jame's account was enabled.
A process has exited.
Special groups have been assigned to a new logon.
Auditing settings on object were changed.
Per Peter Audit Policy was changed.
System audit policy was changed.
Peter's account was logged off.
Peter's account was logged off.
Peter's account was logged off.

Figure 10. Branch event sequences by malicious activity.

The sentence describing the *IOC* generated using logical operators is:

AND (4625, 4625, 4624, 4672, OR (AND (4720, 4728, 4688, 4905, 4722, 4907, 4634)), OR (AND (4740, 4759, 4761, 4698, 4689, 4912, 4634)), OR (AND (4672, 5025, 4688, 4660, 4964, 4719, 4634)))

Figure 12 shows the representation of this sentence using the IOC Editor tool, where the structure of the *IOC* is displayed using simple expressions to create complex expressions using logical operators.

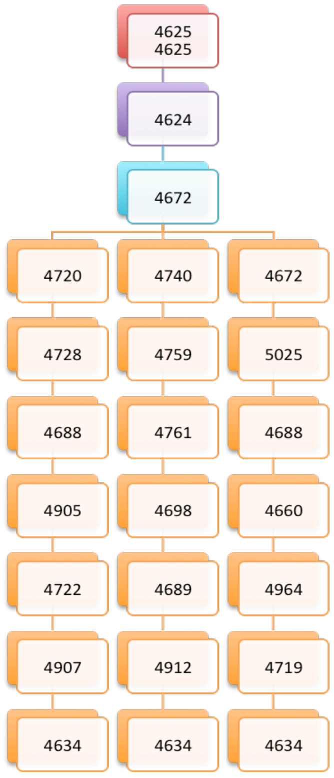
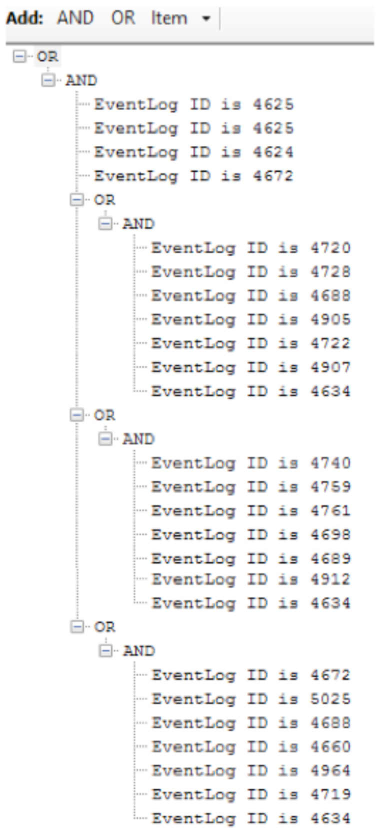


Figure 11. Branch event ID sequence by malicious activity.



```
OR
  AND
    EventLog ID is 4625
    EventLog ID is 4625
    EventLog ID is 4624
    EventLog ID is 4672
    OR
      AND
        EventLog ID is 4720
        EventLog ID is 4728
        EventLog ID is 4688
        EventLog ID is 4905
        EventLog ID is 4722
        EventLog ID is 4907
        EventLog ID is 4634
    OR
      AND
        EventLog ID is 4740
        EventLog ID is 4759
        EventLog ID is 4761
        EventLog ID is 4698
        EventLog ID is 4689
        EventLog ID is 4912
        EventLog ID is 4634
    OR
      AND
        EventLog ID is 4672
        EventLog ID is 5025
        EventLog ID is 4688
        EventLog ID is 4660
        EventLog ID is 4964
        EventLog ID is 4719
        EventLog ID is 4634
```

Figure 12. Indicator of Compromise of complex expression.

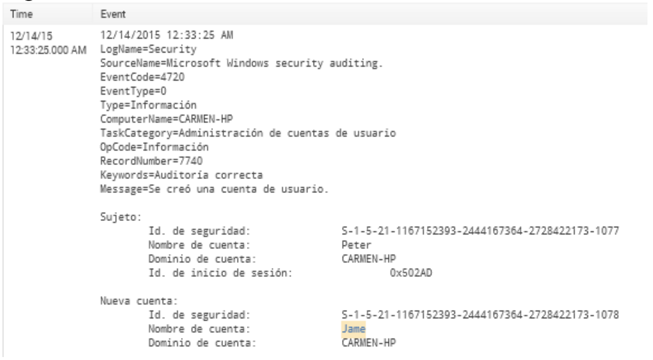
In order to have an overview of the activities during the intrusion on a computer system, the *IOCs* can be generated with the combination of user events, auditing, system and objects.

The general description made above shows the procedure developed to generate *IOCs* using *Windows event logs* to detect intrusions that could result in a security incident on a computer system.

VI. RESULTS

According to the previously described procedure, a list of the most representative *Windows event logs* categories considering how critical the events to generate *IOCs* of complex expressions are; to identify patrons of abnormal behavior that could result in an intrusion, were selected.

An example of a *Windows event log* for a compromised computer system, which describes the creation of the user account "Jame" using Peter´s user account, is shown in Figure 13.



```
Time                Event
12/14/15            12/14/2015 12:33:25 AM
12:33:25.000 AM     LogName=Security
                    SourceName=Microsoft Windows security auditing.
                    EventCode=4720
                    EventType=0
                    Type=Información
                    ComputerName=CARMEN-HP
                    TaskCategory=Administración de cuentas de usuario
                    OpCode=Información
                    RecordNumber=7740
                    Keywords=Auditoría correcta
                    Message=Se creó una cuenta de usuario.

                    Sujeto:
                            Id. de seguridad:              S-1-5-21-1167152393-2444167364-2728422173-1077
                            Nombre de cuenta:              Peter
                            Dominio de cuenta:             CARMEN-HP
                            Id. de inicio de sesión:             0x502AD

                    Nueva cuenta:
                            Id. de seguridad:              S-1-5-21-1167152393-2444167364-2728422173-1078
                            Nombre de cuenta:              Jame
                            Dominio de cuenta:             CARMEN-HP
```

Figure 13. Windows event log structure example.

As previously described in Section V Development, a single *Windows event log* is not sufficient to detect an intrusion; events should relate to other events in the computer system to generate the *IOC* representing that malicious or anomalous activity.

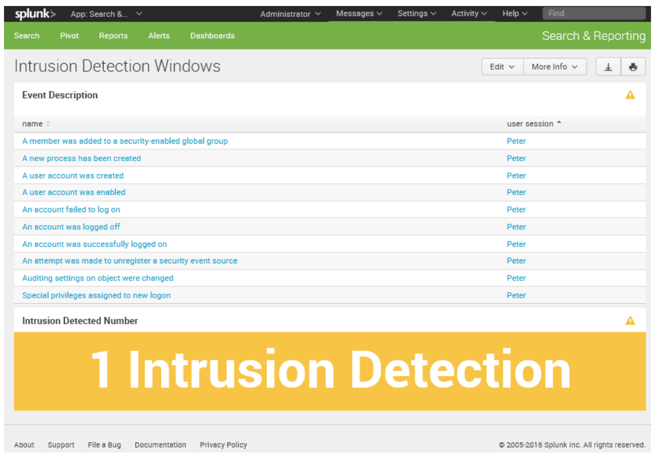
To validate whether the *IOCs* generated are functional or not, these were tested in a correlational tool, for the case study presented above. A free version of the Splunk® tool was used [14], which is a *Security Information and Event Management Software* [9] that allows us to detect intrusion events in real time.

Figure 14 shows an example of an intrusion detection using a generated *IOC*.

In a sampling made on thirty computer systems over an hour, 9996 events were registered, about 50% of these belonging to the categories of User Account Management, Logon and Logoff.

Other registered events that represent 1% are:

- 4735 A security-enabled local group was changed
- 4625 An account failed to log on
- 4738 A user account was changed
- 4634 An account was logged off and
- 4732 A member was added to a security-enabled local group



Intrusion Detection Windows

Event Description

name	user session
A member was added to a security-enabled global group	Peter
A new process has been created	Peter
A user account was created	Peter
A user account was enabled	Peter
An account failed to log on	Peter
An account was logged off	Peter
An account was successfully logged on	Peter
An attempt was made to unregister a security event source	Peter
Auditing settings on object were changed	Peter
Special privileges assigned to new logon	Peter

Intrusion Detected Number

1 Intrusion Detection

Figure 14. Test of intrusion detection using a IOC.

These events are combined with others of higher occurrence or lower occurrence. This helps generate IOCs of simple or complex expressions, which can reduced by more than 80% the rate of false positives.

There are now about 50 *IOCs* to detect intrusion on computers running Windows® 7 OS an up; Table IX shows examples of some of them. They come mainly from forensic analysis.

TABLE IX. EXAMPLES OF IOCS FOR WINDOWS OS.

ID IOC	Logic Description of Events used
1	AND [4618, 4912, OR (4907, 4660,4670,4691), OR (4964, 4767, 4760, 4758, 4757, 4753, 4750, 4743, 4740), OR (5025, 5034, 4950, 4949,)]
2	AND [4649, 4912, 4618, OR (4907, 4660,4670,4691), OR (4964, 4767, 4760, 4758, 4757, 4753, 4750, 4743, 4740), OR (5025, 5034, 4950, 4949,)]
3	AND [4912, 4618, OR (4907, 4660,4670,4691), OR (4964, 4767, 4760, 4758, 4757, 4753, 4750, 4743, 4740), OR (5025, 5034, 4950, 4949,)]
4	AND [4649, 4912, 4618]
5	AND [4618, 4912, 4618]
6	AND [6273, 4618, OR (4907, 4660,4670,4691), OR (4964, 4767, 4760, 4758, 4757, 4753, 4750, 4743, 4740), OR (5025, 5034, 4950, 4949,)]
7	AND [4719, 4618, OR (4907, 4660,4670,4691), OR (4964, 4767, 4760, 4758, 4757, 4753, 4750, 4743, 4740), OR (5025, 5034, 4950, 4949,)]

Most of the generated IOCs are complex expressions and involve more than 15 events of different categories each. 85% of simple expressions can be used to generate different IOCs of complex expressions, which represent different ways that an attack vector may consist of.

The most repeated IOCs events are sometimes considered as non-critical but associated with other events may represent an intrusion or malicious activity.

VII. CONCLUSION AND FUTURE WORK

To summarize, *IOCs* can be generated by the combination and sequence of different *Windows event logs*, describing a possible malicious activity on a computer system, adding them to the techniques known to generate *IOCs*, which provides to *CIRTs*, another way to detect an intrusion in a computer systems.

The use of *IOCs* with *Windows event logs* improves *IR* and forensic analysis processes, allowing to know the activities of an intruder on a computer system, and managing to make proper actions to prevent future malicious activities with the same attack vector in the system.

As future work, we propose to analyze what other events can be added to improve the *IOCs* generated and perform the same procedure on other OS such as Linux.

ACKNOWLEDGMENT

Thanks to Instituto Politécnico Nacional for the support granted during the development of this research.

REFERENCES

[1] <http://searchsecurity.techtarget.com/definition/Indicators-of-Compromise-IOC> 2015.09.23

[2] P. Kral, “The Incident Handlers Handbook,” SANS Institute InfoSec Reading Room, 2011.

[3] T. Sager, “Killing Advanced Threats in Their Tracks: An Intelligent Approach to Attack Prevention,” SANS Institute InfoSec Reading Room, 2014.

[4] <https://technet.microsoft.com/en-us/library/dn487458.aspx> 2015.11.07

[5] H. Lock, “Using IOC (Indicators of Compromise) in Malware Forensics,” SANS Institute InfoSec Reading Room, 2013.

[6] <https://www.auscert.org.au/render.html?it=4323&template=1> 2015.12.16

[7] <https://www.fireeye.com/content/dam/fireeye-www/services/freeware/ug-ioc-editor.pdf> 2015.09.25

[8] A. Chuvakin and L. Zeltser, “Critical Log Review Checklist For Security Incidents,” Cheat sheet version 1.0, SANS.

[9] K. Kent and M. Souppaya, “Guide to Computer Security Log Management, NIST Special Publication 800-92,” Recommendations of the National Institute of Standards and Technology, USA 2006.

[10] <http://searchwindowsserver.techtarget.com/definition/Windows-event-log> 2016.01.04

[11] <https://technet.microsoft.com/en-us/library/cc722385%28v=ws.10%29.aspx> 2015.11.10

[12] <http://openioc.org/resources/An_Introduction_to_OpenIOC.pdf> 2015.12.05

[13] <https://technet.microsoft.com/en-us/library/dn535498.aspx> 2015.11.12

[14] <http://www.splunk.com/> 2015.10.10

Comparative Study of Routing Protocols in Ring Topologies using GNS3

Roberto Alejandro Larrea-Luzuriaga[1], Jose Miguel Jimenez[2], Sandra Sendra[2], Jaime Lloret[2]
[1]Universidad Politécnica de Valencia, Valencia (Spain).
[2]Instituto de Investigación para la Gestión Integrada de zonas Costeras, Universidad Politécnica de Valencia, Grao de Gandia (Spain)
e-mail: rolarlu@teleco.upv.es, jojiher@dcom.upv.es, sansenco@posgrado.upv.es, jlloret@dcom.upv.es

***Summary*— Routing is the function of seeking a path between all possible in a packet network topologies which have great connectivity. Because it comes to find the best possible route, the first step is to define what constitutes best route and consequently what is the metric to measure it. This parameter and the operation of the routing protocol itself give us the protocol performance. This work presents a comparative study of three of the most used routing protocols, i.e., Routing Information Protocol (RIP), Open Shortest Path First (OSPF) and Enhanced Interior Gateway Routing Protocol (EIGRP) in ring topologies. Through this study, the performance of each routing protocol is analyzed. To this purpose, we have used a network simulator known as GNS3 that allows simulating different network scenarios using real operating systems (IOS) of CISCO equipment. We have simulated a topology with these three protocols in order to observe and analyze the network behaviour, traffic flow, time of routes updating and network convergence. As results show, RIP presents the lowest time for initializing the network while OSPF is the protocol that presents highest time in initializing process of network. Finally, EIGRP shows the best convergence time after the first fault in the network.**

Keywords- Routing Protocol; Convergence time; RIP; OSPF; EIGRP; GNS3; Network Performance.

I. INTRODUCCIÓN

Routing is the process through which a router determines the best route of a data packet to reach a destination. This packet passes through several devices so that the network destination is different to the origin network. There are two types of routes, i.e., statics and dynamics routes. Static routes are those that are configured by hand or are specified by default and do not have any reaction to new routes or falling sections of the network routes. However, a router with dynamic routing is able to understand the network and pass routes between neighbouring routers. The network through routers with dynamic routing is responsible for specifying the access to new nodes on the network or adapts and modifies the access to certain parts of the network due to the fall of any link or node seeking an alternative optimal route [1].

Routing protocols are algorithms that allow to determine and to select the best route upon which the network traffic will be send from one network to another. To this end, these algorithms use different information associated to links, such as bandwidth, delay, load, reliability, number of hops or cost, among others [2]. In this way, the exchange of information between the equipment can generate the existing network topology and determine the best links to be used to reach a specific destination.

Routing protocols are subdivided in two types, distance vector and link state. On the one hand distance vector algorithms use the Bellman-Ford algorithm. It searches the path of lowest cost by the indirect method search. The distance vector associated to a network node is a control packet that contains the distance to the nodes of the network known so far. Each node sends its neighbours the distances that knows through this packet. The neighbouring nodes examine this information and compare it with the information they already have updating its routing table, if necessary. Some examples of distance-vector protocols are Routing Information Protocol (RIP) (version 1 and 2), Interior Gateway Routing Protocol (IGRP) and Enhanced Interior Gateway Routing Protocol (EIGRP). On the other hand, link status algorithms are based on each node gets to know the network topology and costs (delays) associated with the links. From these data, nodes can obtain the tree and the routing table after apply the minimum cost algorithm (Dijkstra algorithm). Open Shortest Path First (OSPF) and Intermediate system to intermediate system (IS-IS) protocols are examples of link state protocols.

Because of the complex operation of these algorithms, it is necessary understanding the protocols operation in order to select the more adequate protocol for the designed network. Additionally to the basic operation, researchers usually modify some of these features in order to provide improvements in energy consumption in the transmission of information [3] [4] and the own consumption of network devices [5].

On the other hand, it is important to know the most appropriate sort of topology that our network needs. We can find Simple topologies such as bus, star or ring topology and more complex structures as mesh networks. Besides these topologies, they can be combined for grouping nodes in clusters [6] or in groups [7].

Networks with ring topologies offer one grade of redundancy to a possible fail so that if one link fails, it is possible to maintain the communication between all network nodes. This fact is essential in backbone networks [8]. In a backbone network with ring topology where dynamic routing with default values is configured, every node of the ring, after the network has converged, has a data base with the information about the network routes toward every node, where there will be information about how to arrive to the destination for two routes with equal cost. It is known as redundancy.

The time that a routing protocol takes to calculate the route to achieve the destination, as well as the convergence time to start data transmission after a failure and recovery time are different depending on the protocol and the mode that it exchanges information about the network topology. For this reason, some protocols will have better performance than another.

Taking into account all of these issues, in this paper, we are going to perform a comparative analysis with three routing protocols, RIP, OSPF and EIGRP, in order to check which one offer the best performance when a ring topology is used. To do this, we are going to use a network simulator called GNS3 [9]. These protocols are used in a ring topology composed by 5 routers and a computer that monitors all events. The tests are performed in terms of convergence time when network begins to work, recovery time after a failure and network response when one or two links register a failure.

The rest of this paper is structured as follows. Section 2 shows some previous studies where these protocols have been analyzed. Section 3 presents the protocols RIP, OSPF, and EIGRP that we have used in our test bench and simulations. The simulation scenario and some previous measurements are shown in Section 4. Section 5 shows the result of our study. Finally, Section 6 presents the conclusion and future work.

II. Related Work

There is a big number of previous works where routing protocols are studied and evaluated using network simulators. There are several network simulators which are widely used in telecommunications such as OPNET, OMNET, NS-2 or NS-3, among others. Some of them are free and other ones can only be used under license. In any case, the most important thing is that these simulators should offer results as realistic as possible [10]. This Section shows some of these works.

G. S. Aujla and S. S. Kang [11] performed a study and analysis of several routing protocols over Mobile ad hoc networks (MANETs). Authors used OPNET simulation to test the operation of five well known routing protocols as Ad Hoc On-Demand Distance Vector (AODV), Dynamic Source Routing (DSR), Temporally-Ordered Routing Algorithm (TORA), Optimized Link State Routing (OLSR) and Geographic Routing Protocol (GRP). The protocols performance was measured on the basis of throughput, delay, load and data dropped metrics. In addition, the work was focussed on the e-mail and video conferencing traffic generating applications by increasing the number of nodes. As a result of this study, we can see that AODV is the one which present better results for video conferencing when a low number of nodes. However, OLSR protocol shows good results for email traffic

K. Yao et al. [12] presented a real-time testbed for routing network (ARTNet) in order to evaluate the requirements that routing protocols should present. As authors state ARTNet supports some of the most popular routing protocols used in typical applications in a cost-effective way. To test the good operation of ARTNet, authors have performed several simulations with two popular routing protocols, EIGRP and OSPF based on quantitative metrics, i.e., packet loss and delta time for standard application services. The results of this study demonstrate that EIGRP presents converge time faster for HTTP and FTP applications when the primary link for a subnet suffers a fault. This kind of simulators and proposal are a good tool that allows users to easily make different network configurations.

M. I. Ashraf et al. [13] presented a comparative analysis of OSPF and EIGRP. Authors study the network performance of these protocols in enterprise environments. The study was performed at simulation level using OPNET Modeler. Along the paper, authors explained the main features that a protocol should present in order to be used in big networks that manage the information in corporative and enterprise environments. The study evaluated OSPF and EIGRP performance in terms of convergence time, scalability and resources Utilization through the simulated network models. As results shows, EIGRP seems to be more scalable in terms of routing domain size. On the other hand, OSPF is more efficient in terms of router and CPU utilization, message processing, routing table size and the traffic load through the network. In terms of convergence time, EIGRP is faster than OSPF. This report can be understood as a user guide to use EIGRP, OSPF in certain circumstances.

There are several comparative studies where more complex topologies have been studied, but we have not found studies of routing protocols in ring topologies. However, there are interesting proposals based on ring topologies such as [14] where authors presented a set of test benches to study the TCP/IP interaction based on congestion price for evaluating the stability and optimality in ring topologies and [15] where X. Li describes a particle swarm optimization (PSO) algorithm using a ring neighborhood topology, which does not require any niching parameters. For this reason and because of the features of this kind of topologies, we have decided to perform this work.

III. Routing protocols

This section presents some of the most widely used routing protocols. For each protocol, it is explained the main characteristics as well as default values of each parameter that these protocols use when they are running.

A. RIP and RIP V2

RIP [16] is an Interior Gateway Protocol (IGP) used by routers to exchange information about IP networks that are connected. This protocol is based on the distance vector algorithm and it uses the User Datagram Protocol (UDP) to send information through the network. The router that uses this protocol has a limited knowledge about the network information. This protocol uses the hops number mechanism to determine the best route. It supports up to 15 hops to avoid routing loops. A route with 16 hops is considered a route as unreachable or not desirable. This fact limits the network size. It is a popular protocol due its simplicity and easy configuration. However, the main problem lies in the convergence times and scalability limitations, so it has a better performance in small networks [17].

On the other hand, RIPV2 [18] was created due to the necessity of supporting variable length subnet masks (VLSM) and other requirements enhanced in respect to its first version. In addition to including VLSM, RIPV2 supports the process of routes authentication and it incorporates the routes updating making use of multicast packets, classless inter-domain routing (CIDR) and the updating by trigger. It keeps the sending of updating packets each 30 seconds and a limit of 15 hops. It uses the same port UDP, i.e., the port 520, and it uses the strategies of inverse poisoning and counting to infinite to prevent loops, as the first version of RIP [19].

Unlike other protocols, RIP is a free protocol to be used by different routers and not only by a single one owner such as EIGRP which was developed by Cisco Systems. Table I [20] shows a summary of the default parameters of RIP protocol.

TABLE I. PARAMETERS OF RIP

Parameter	**Default values of RIP Parameters**	
	Description	***Value***
Updating interval	Time period to sending actualizations, to its neighbours.	30 s.
Invalid route	It is a initialized timer when a route is inserted in the routing table. When this time expires the route is invalid.	180 s.
Flush	It marks that a route must be removed of routing table. This value must be bigger that the value to the invalid route.	240 s.
Holddown	It is used to avoid that one route has been marked as valid immediately after of it had marked as invalid. During this time the actualizations respect to the invalid route are ignored.	180 s.
Announcing's methods	It specifies the mechanisms that the router uses to communicate with its neighbours. 1- No filtering: Announce the routes for all its neighbours. 2- Split horizon: Do not announce a route to one neighbour from which it was learned. 3- Split horizon with poison reverse: Announce the route to the neighbour of which it was learned with a metric to infinite or 16 maximum.	3

B. *OSPF*

OSPF [21] is a link state routing protocol that uses the algorithm short path first (SPF) to calculate the best route to the destination node. It is one of the most widely used hierarchical protocols due to its significant scalability.

OSPF keeps all network information in its routing table. For this reason, it requires a major level of processing and memory. The header of OSPF packets includes the source and destination address. OSPF uses multicast as destination address and sends many message types including hello messages, link state requests and updates and database descriptions. Djisktra's algorithm is used to specify the shortest path to the destination. SPF calculations are computed either periodically or upon a received Link State Advertisement (LSA). This fact depends on the protocol implementation. The protocol operation is based on 5 different types of link states packets (LSP's). These packets help the protocol to distinguish between its neighbours and update the routing information of link states. The routers of a network that runs OSPF can have different roles as a function of its position. These roles are internal router, backbone router, edge area router and autonomous systems (AS) router. OSPF uses the accumulated bandwidth as metric from the source interface to destination interface to calculate the cost. Finally, it is important to know that OSPF supports VLSM [17]. Contrary to RIP, however, OSPF has the disadvantage of being too complicated. Table II shows a summary of the main parameters of OSPF protocol [20].

TABLE II. PARAMETERS OF OSPF

Parameter	**Default values of OSPF parameters**	
	Description	***Value***
Cost Interface	The cost of each interface can be specified, this parameter is used to use the short path first.	1
Hello's messages interval	Time period to send Hello's messages to its neighbours. If this parameter is too small, result in more traffic to the router, that it increments the risk of that the packets are discarded, so it could producing false alarms. If the value is too big, the change detection times in the topology are majors, the router dead timer could be expire.	10 s.
Router dead timer	Timer used, to declare to its neighbours as down, when the Hello messages didn't have been received. Its value must be multiple of the Hello interval.	40 s.
Transmissi on delay	It is the estimated time to transmit notification packets about link state LSA.	1 s.
Retransmis sion interval	Retransmission time LSA. It must be major that the round trip time estimated between any couple routers in the network.	5 s.
Parameters to the calculate SFP	It specifies how often it calculates the short path first: 1- Periodic: It recalculated in each specified interval, unless it hasn't occurred any change. 2- LSA delivered: It recalculated after of each LSA has been received.	2

C. *EIGRP*

EIGRP [22] is an advanced routing protocol based on distance vector, owned by Cisco Systems, which offers the best characteristics of both distance vector and link state algorithms. This protocol can only be used in Cisco routers. It is a fast convergence routing protocol and extremely scalable for medium and big networks. In addition, EIGRP implements CIDR and VLSM. It is considered an advanced protocol because it is based on features commonly associated with link-state protocols and although it may serve as it, the fact is that EIGRP is a distance vector routing protocol. EIGRP uses some of the best features of OSPF, such as partial updates and neighbor discovery.

The key of its good performance is use of the Diffusing Update ALgorithm (DUAL) for updating routes. Through it, this protocol achieves an exceptional and rapid network convergence. The metrics takes into account the bandwidth, load, feasibility and delay. It keeps the routes' information and topology's network details in three different tables called neighbour table, topology table and routing table [4]. This algorithm uses the neighbour table and the topology table to develop the routing table in the EIGRP router. The topology table is created by the finite states machine of DUAL using the collected information of its routers neighbours. With the available information of the topology table, DUAL calculates the best route to the router destination and it makes that link as successor. DUAL also calculates the feasible successor (second best link free of loops) if it is available. Although EIGRP does not guarantee the use of the best route, this protocol is rather used because it is easier to configure than OSPF.

Table III shows the main parameters used by EIGRP protocol [20].

TABLE III. PARAMETERS OF EIGRP

Parameter	**Default values of EIGRP parameters**	
	Description	***Value***
Hello interval messages	Period of time to send Hello messages to its neighbours. If this parameter is too small, the result is more traffic to the router, so it increases the risk of that packets are discarded, so it could producing false alarms. If the value is too big, the changes detection time in the topology will be major, the router dead timer could expire.	5 s.
Hold Time	Timer used to declare to the neighbours as down, when the Hello messages has not been received. Its value must be multiple of Hello interval.	15 s.
Split Horizon	When it is enabled, it does not notifies the routes to the neighbours of which were learned.	Enable

As we can see, the EIGRP is simpler in terms of the number of different types of packets generated by the protocol.

IV. SIMULATION ENVIROMENT

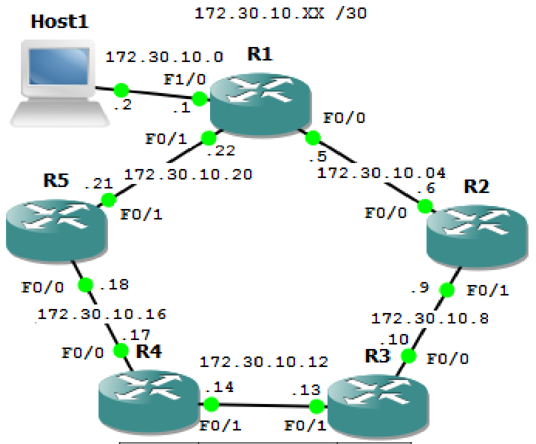
In order to evaluate the performance of each protocol, we have implemented a basic ring topology. This topology and response times of each router are shown in this Section.

As Figure 1 shows, the topology is composed by 5 routers linked by Fast Ethernet interfaces. In this case, the devices used are Cisco routers with the Cisco IOS Release 12.4(13b) for 2691 service Platform with IP Base. To perform the simulations, a personal computer (PC) is connected to the ring topology. Each router has, at least, 2 network interfaces except the router R1 which needs one more interface to connect the PC. This PC is in charge of monitoring all network events making use of the management software. Figure 1 also shows the subnet network and the physical interfaces that connects each router.

To design the network addressing, we have used VLSM. We have defined 5 subnets with subnet mask /30 starting from the class B address 172.30.10.0/16.

In order to see the initial performance of this network, we have configured static routes in each router. After that, a ping is transmitted from Host 1 to each router following the route with lowest cost. Figure 2 shows the Round Trip Time (RTT) in ms. where routers are configured with static routes. With the value of RTT, it is easy to measure the time during which part of the network does not have connectivity. We can also compare the times for symmetric routes.

From these results, we can calculate the average response time to reach each router (See Figure 3). Figure 3 also shows the percentage of packet loss. In this case, because we are working static routes and a very simple network, we do not register packet losses. If results are analyzed, we can observe that the biggest response time is registered by the farthest network to R1, i.e., the route up to network 172.30.10.12/30 through the interface F0/1 of router R3. We can observe that the minimum response time is established to reach the router R1 in 21 ms, and the maximum time is 64 ms to reach the node R3. In addition, it is important to see that this value is doubled when two hopes are needed to reach the destination.



Device	Subnet	Device
PC	172.30.10.0/30	R1
R1	172.30.10.4/30	R2
R2	172.30.10.8/30	R3
R3	172.30.10.12/30	R4
R4	172.30.10.16/30	R5
R5	172.30.10.20/30	R1

Figure 1. Network in ring topology used in our simulations.

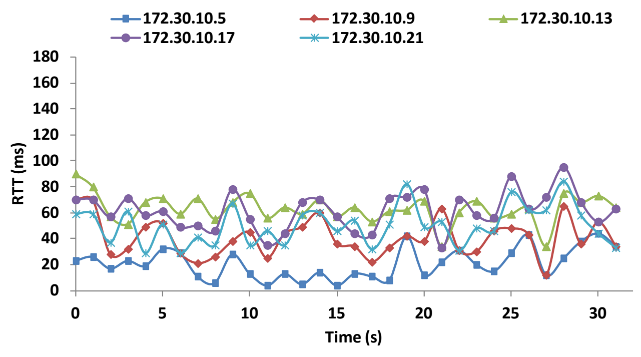


Figure 2. Response times from Host 1 to all routers

It is easy to see that symmetric routes, such as PC to R5 and PC to R2, present a difference of average response time of 10ms. If we analyze both, route from PC to R3 and route from PC to R4 this difference between the average response times is lower.

V. PROTOCOLS' EVALUATION

This Section presents several tests where RIP, OSPF and EIGRP are evaluated. Protocols are evaluated in terms of recovery time after a network failure. This Section also analyzes the different states that a network running a routing protocol can suffer, i.e., event of updating messages, event of network initialization, network failure and network recovery.

When configuring the routing protocols with its default parameters, routers calculate 4 optimal routes to reach the no adjacent networks from each router. Because we have a ring topology, each router only has two optimal routes to reach the farthest network to each node. To evaluate the performance of each routing protocol, we need to determine the learning and route publication time and the ring network convergence, in three different stages: (1) at the beginning of network operation, (2) when a dropped link is registered and (3) when this link is restored. The failure point has been established in the link between router R1 and R5. When the network starts, routers have to calculate and learn the two optimal routes to reach the two non adjacent networks in both directions through the interfaces of each router with a same cost. To determine the recalculating times of routes and the network convergence as a result of a dropped link, the monitoring point is established in the farthest network from router R5, i.e., Host 1 is used to monitor the network activity. The failure point is established in the fast Ethernet interface 0/1 (F0/1) of router R5. After generating the failure, the link is reconnected and restored. The time elapsed between the failure generation and the communication re-establishment will give us the time of network inactivity time which will consider the convergence time and some additions milliseconds that the devices will need to route the packets. As packets, we have used pings with the parameters by default.

A. Evaluation of RIP

In order to evaluate the network performance when RIP is running, we have generated a failure in the link between the R1 and R5. After that, the link is restored. To evaluate the restoring time of this protocol, we have sent a continuous ping between Host 1 and each router. Figure 4 shows the RTT in ms. of each ping between Host 1 and each router. As we showed in Figure 1, R4 and R5 are reached through the shortest path, i.e., R4 is reached through R5. As Fig. 4 shows, the disconnection and restoration of link is generated at 25th second. From this moment, the network needs around 475 s to recover the communication with R4 and R5. We can also observe that R1, R2 and R3 have not lost the connectivity with Host 1 although the RTT of their ping has slightly increased. This is because the protocol needs to inform to the rest of routers about the new path to reach R4 and R5. The RTT for R4 and R5 has increased about 110 milliseconds.

Table IV shows the response times evaluated for this protocol, in the different phases of its operation.

TABLE IV. RESULTS OF RIP

Event	RIP Results	
	Action	*Time (s)*
Updating messages	--	29.601
Network start	Learning and publishing routes	7.515
	Ring network convergence	13.823
Failure	Learning and publishing routes	27.931
	Ring network convergence	298.129
Recovery	Learning and publishing routes	2.269
	Ring network convergence	30.888

B. Evaluation of OSPF

In order to evaluate the restoring time of OSPF, we have sent a continuous ping between Host 1 and each router. In this case we have generated 2 failures in the same link. The second one is generated after restoring the network communication of the first failure. Fig. 5 shows the RTT in ms. of each ping between Host 1 and each router after the first failure. Figure 5 shows that the disconnection and restoration of link is generated at 7th second. From this moment, the network needs around 50 s to recover the communication with R5 and around 10 s to recover the communication with R4. We can also observe that R1, R2 and R3 have not lost the connectivity with Host 1. However, the RTT of their ping has increased around 10-15 ms. After restoring the communications with R4 and R5, the RTT is around 110 ms for R4 and 140 ms for R5.

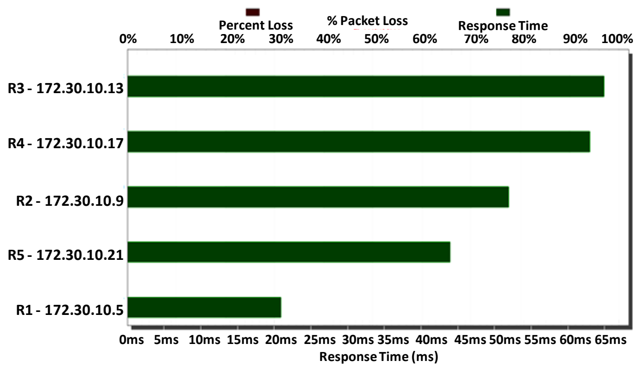


Figure 3. Average response time

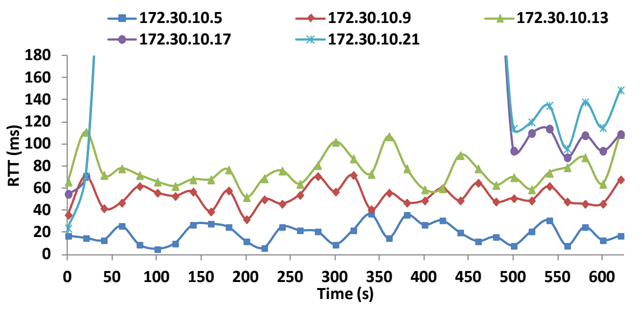


Figure 4. RTT and communication restoring time for RIP after a failure

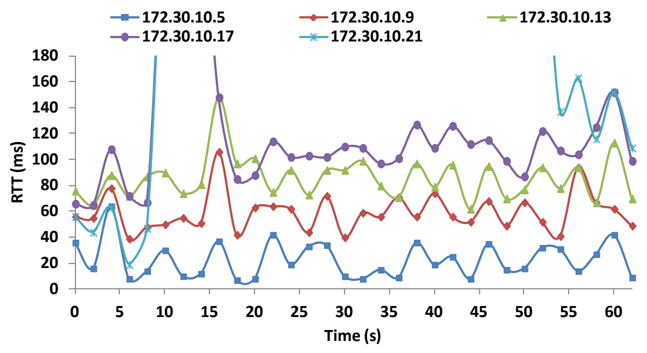


Figure 5. RTT and communication restoring time for OSPF after the first failure

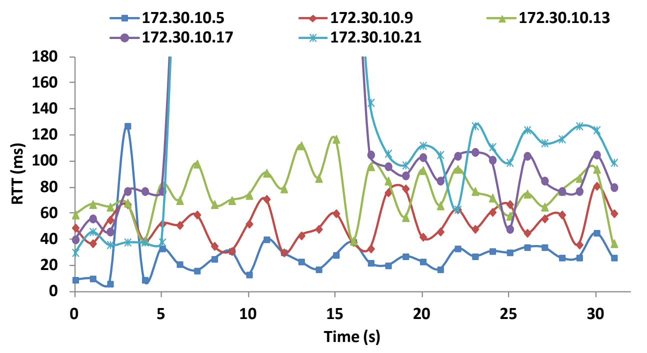


Figure 7. EIGRP data convergence before the failure

After generating the second fault and restoring it in the same link (in 15th second), we have observed that the network only needs 7 seconds to establish the connectivity (See Figure 6). This is because the routing tables already contain the alternative route to reach R4 and R5. The RTT values are similar to Figure 5.

Table V shows the response times registered for OSPF protocol, in the different stages.

TABLE V. RESULTS OF OSPF

Event	OSPF Results		
	Action	*Time (s)*	
Hello messages	--	9.99	
Network start	Learning and publishing routes	53.84	
	Ring network convergence	66.78	
Failure	Number of failure of the same link	1st	2nd
	Learning and publishing routes	42.14	2.51
	Ring network convergence	45.00	5.62
Recovery	Learning and publishing routes	7.76	
	Ring network convergence	14.80	

As Table V shows, the response times improve after the second failure. This is because the routing tables maintain the information about how to reach R4 and R5. The calculation of alternative routes is faster than the previous situation. The elapsed time without communication is the time that routers need to switch to the alternative route.

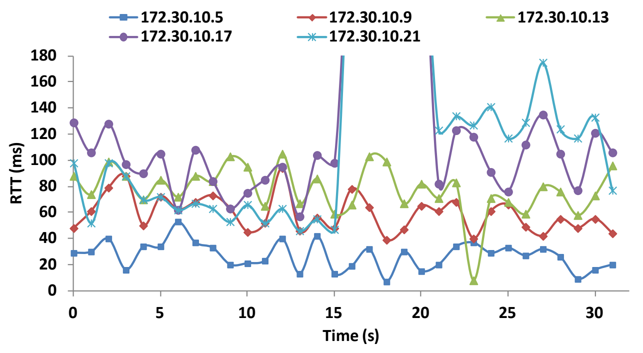


Figure 6. RTT and communication restoring time for OSPF after the second failure

C. *Evaluation of EIGRP*

To evaluate the network performance when EIGRP is running, we have generated an only failure in the link between the R1 and R5. Figure 7 shows the RTT in ms. of each ping between Host 1 and each router. The disconnection and restoration of link is generated at 5th second. From this moment, the network needs around 12 s to recover the communication from Host 1 to R4 and R5. R1, R2 and R3 have not lost the connectivity with Host 1. The RTT of pings for R1, R2 and R3 are around 28 ms, 52 ms. and 75 ms, respectively. The average value of RTT for R4 is about 91 ms and for R5 is 111ms.

Finally, Table VI shows the different response times for EIGRP protocol in its different stages it needs to correctly work.

TABLE VI. RESULTS OF EIGRP

Event	EIGRP Results	
	Action	*Time (s)*
Hello messages	--	4.939
Network start	Learning and publishing routes	6.677
	Ring network convergence	15.241
Failure	Learning and publishing routes	0.9547
	Ring network convergence	12.573
Recovery	Learning and publishing routes	1.492
	Ring network convergence	1.965

As observed, EiGRP presents the lowest times compared to RIP and OSPF.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have analyzed the network performance of ring topology when RIP, OSPF and EIGRP protocols are executed.

After evaluating these routing protocols, we should highlight several aspects of ring topologies as a function of the routing protocol it runs. On the one hand, RIP presents a good response time at the beginning of the network activity while its performance decreases after a failure. RIP requires higher time in the network establishing, when the network is recovered from the failure.

On the hand, OSPF has a higher time in the network starting, but unlike of RIP, its performance improves after the first failure. It also improves its convergence time, even when a second failure is registered in the same link. OSPF registers better times response that the ones registered by EIGRP.

The simulations shows that after a failure the ring routers converge in an asynchronous form, i.e., the connectivity with the other routers is restoring according to time that each router needs to calculate the new optimal route.

EIGRP recorded the best convergence times of the ring after a failure and it also presents the best RTT when it is restored.

As future work, we would like to extend the comparative analysis of routing protocols for mesh topologies [23] in order to check their behaviour after a failure and the impact of this fact over the network efficiency. In addition, we have also found interesting approaches related to genetic algorithms based on ring topologies [24] and we would like to explore this possibility to improve the efficiency of sensor networks.

ACKNOWLEDGMENT

This work has been supported by the “Ministerio de Economía y Competitividad”, through the “Convocatoria 2014. Proyectos I+D -Programa Estatal de Investigación Científica y Técnica de Excelencia” in the “Subprograma Estatal de Generación de Conocimiento”, project TIN2014-57991-C3-1-P, by the “Contratos Postdoctorales UPV 2014 (PAID-10-14)” by the“Universitat Politècnica de València” and by the “Programa para la Formación de Personal Investigador – (FPI-2015-S2-884)” by the “Universitat Politècnica de València”

REFERENCES

[1] S. Sendra, P. Fernández, M. Quilez, and J. Lloret, “Study and Performance of Interior Gateway IP Routing Protocols”, Integrated Management Coastal Research Institute, Polytechnic University of Valencia, Network Protocols and Algorithms, 2010, Vol. 2, No. 4, pp. 88-117.

[2] J. Deng, S. Wu, and K. Sun, “Comparison of RIP, OSPF and EIGRP Routing Protocols based on OPNET,” Simon Fraser University School of Engineering Science. ENSC 427: Communication Networks, 2014.

[3] Q. Ling, J. Yan, and H. Deng, “A Novel Energy-Aware Routing Algorithm for Wireless Sensor Networks Based on CDMA and TDMA,” Ad Hoc & Sensor Wireless Networks, 2015, Vol. 26, No. 1-4, pp. 21-41.

[4] M. Eslaminejad, S. A. Razak, and M. Sookhak, “Classification of Energy-Efficient Routing Protocols for Wireless Sensor Networks,” Ad Hoc & Sensor Wireless Networks, 2013, Vol. 17, No, 1-2, pp. 103-129

[5] S.Andrade, S. Sendra, E. Granell, and J. Lloret, “Towards green networks using optimized network devices.” (Ch. 15), Green Networking and Communications: ICT for Sustainability,2013, CRC Press, Taylor & Francis Group, pp. 355-376.

[6] M, Atto and C. Guy, “Routing Protocols for Structural Health Monitoring of Bridges Using Wireless Sensor Networks,” Network protocols and Algorithms, 2015, Vol 7, No 1, pp.1-23.

[7] J. Lloret, S. Sendra, M. Garcia, and G. Lloret, "Group-based underwater wireless sensor network for marine fish farms," 2011 IEEE GLOBECOM Workshops, Houston, Texas, USA, December 5-9, 2011. pp.115,119.

[8] B. Meador. “A Survey of Computer Network Topology and Analysis Examples,” Academic Report. In Washington University website. Available at http://www.cse.wustl.edu/~jain/cse567-08/ftp/topology/#ring_network_topology [Last Access: January 25, 2016]

[9] C. Welsh, “GNS3 Network Simulation Guide,” Packt Publishing, ISBN 13 9781782160809, October 2013

[10] A. Basu et al., “The state of peer-to-peer network simulators,” ACM Computing Surveys (CSUR), 2013, Vol. 45, No. 4, pp. 1-25.

[11] G. S. Aujla and S. S. Kang, “Comprehensive Evaluation of AODV, DSR, GRP, OLSR and TORA Routing Protocols with varying number of nodes and traffic applications over MANETs,” IOSR Journal of Computer Engineering (IOSR-JCE), 2013, Vol. 9, No. 3, pp 54 -61.

[12] K. Yao, W. Sun, M. Alam, M. Xu, and V. Devabhaktuni, “A Real-Time Testbed for Routing Network,” In proceedings of the 8th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM 2012), Thessaloniki, Greece, June 11-13, 2012. pp. 256-270.

[13] I. Ashraf, S. Iftikhar, U. Sarwar, and A. Latif, “Comparative Analysis of Link State and Hybrid Routing Protocols,” International Journal of Computer Science and Management Research, 2013, Vol. 2, No. 4, pp. 2244-2255.

[14] J. He, M. Chiang and,J. Rexford, "TCP/IP interaction based on congestion price: Stability and optimality." In proceedings of the IEEE 2006 International Conference on Communications (ICC 2006). June 11-15, 2006, Istanbul, Turkey. pp. 1032-1039.

[15] X. Li, "Niching without niching parameters: particle swarm optimization using a ring topology," IEEE Transactions on Evolutionary Computation, 2010, Vol. 14, No. 1, pp. 150-169.

[16] RIP Features. In IETF Website. Available at: https://tools.ietf.org/html/rfc1058 [Last Access: January 25, 2016]

[17] D. Sankar and D. Lancaster, “Routing Protocol Convergence Comparison using Simulation and Real Equipment,” Advances in Communications, Computing, Networks and Security Volume 10, ISBN: 978-1-84102-358-8, pp186-194, 2013.

[18] RIP V2 Features. In IETF Website. Available at: http://tools.ietf.org/html/rfc1723[Last Access: January 25, 2016]

[19] A. Bruno and J. Kim., “CCDA Self-Study: RIP, IGRP, and EIGRP Characteristics and Design,” Cisco Press , Pearson Education 2015.

[20] M. Nguyen, K. Mirzahossein, and S. Elmasry, “Analysis of RIP, OSPF, and EIGRP Routing Protocols using OPNET,” Simon Fraser University School of Engineering Science ENSC 427: Communication Networks, 2013.

[21] OSPF Features. In IETF Website. Available at: https://tools.ietf.org/html/rfc1247 [Last Access: January 25, 2016]

[22] EIGRP Features. In IETF Website. Available at: http://tools.ietf.org/html/draft-savage-eigrp-02 [Last Access: January 25, 2016]

[23] D. Lee, K. Lee, S. Yoo and, J.-K. K. Rhee, "Efficient Ethernet Ring Mesh Network Design", Journal of Lightwave Technology, 2011, Vol. 29, No. 18, pp. 2677-2683.

[24] N. Tian, J. Sun, W. Xu and, C.-H. Lai, “Quantum-Behaved Particle Swarm Optimization with Ring Topology and Its Application in Estimating Temperature-Dependent Thermal Conductivity,” Numerical Heat Transfer, Part B: Fundamentals, 2011, Vol. 60, No. 2, pp. 73-95.

Enhancing Network Security Environment by Empowering Modeling and Simulation Strategy

(Cyber Protect Simulation Lesson Learned)

Rudy Agus Gemilang Gultom
Deputy for Strategic Studies
The National Resilience Institute of the Republic of Indonesia (Lemhannas RI)
Jakarta, Indonesia
e-mail: rudygultom@lemhannas.go.id

Baskoro Alrianto
Bureau of Telematics
The National Resilience Institute of the Republic of Indonesia (Lemhannas RI)
Jakarta, Indonesia
e-mail: karotelematika@lemhannas.go.id

***Abstract*—This paper provides an overview based on cyber protect simulation experiences to enhance network security environment by empowering modeling and simulation strategy. Cyber protect is a simulation tool developed by the US Defense Information Systems Agency (DISA). Cyber protect simulation is an integral part of cyber security for information leaders course at National Defense University (NDU), Washington, DC. USA. Strategic thoughts can be implemented during cyber protect simulation exercises. Brilliant ideas in simulating an organization network security environment become good lesson learned. The implementation for proper defense strategy could secure an organization Local Area Network (LAN) from various threats, attacks and vulnerabilities in concrete and abstract levels. Countermeasure strategy, which is implemented in this simulation exercise is presented as well. At the end of this paper, an initial network security framework concept, so called The Six-ware Framework concept (The SWF concept) has been introduced.**

Keywords-cyber protect simulation; threats, attacks and vulnerabilities; countermeasures strategy; network security framework and models

I. Introduction

Nowadays, as the cost of information processing and internet accessibility falls, civilian, military and government organizations security environments are becoming increasingly vulnerable from cyber threats or attacks, e.g., network intrusions, DoS/DDoS, phishing, spoofing, viruses, flooding, etc. At this point, the information security manager might allocate budget, spreading it for network defense tools, e.g., anti-virus software, firewalls, intelligent routers or expensive modeling and simulation (M&S) tools. M&S is an effective technique to support better understanding for information security managers in concrete and abstract levels [1]. M&S can be used to identify weaknesses proactively and it can also provide education and training using "what if" scenarios reactively. Ultimately when new threats appear the ability of an organization to respond is significantly enhanced.

One good lesson learned in the context of information security issue today is the phenomenon of Panama papers where over 11.5 million files have been leaked including 2.6 terabytes of data. E-mails accounted for the majority of exposed records (4,804,618 files), followed by database formats (3,047,306), PDFs (2,154,264), images (1,117,026), text documents (320,166) and other (2,242) files (see Figure 1). At this point it is still unclear whether the 11.5 million files were obtained through hacking (data breach) or leaked from someone inside of the Panamanian law firm (insider leak). But from a cyber protect perspective, the lessons are nearly identical either way [2][3][4][5].

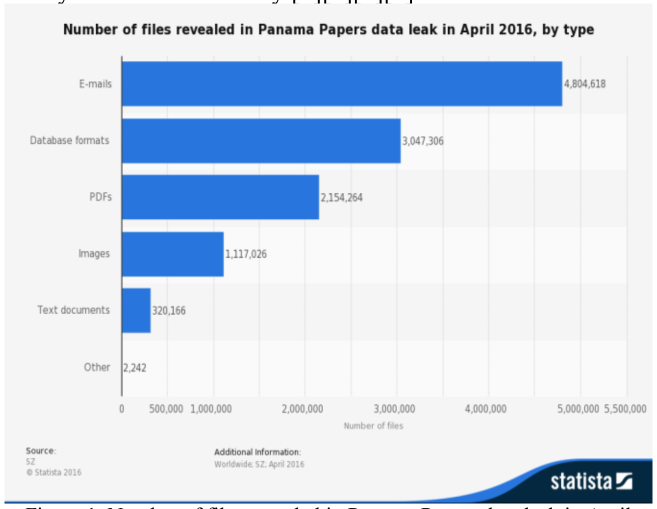


Figure 1. Number of files revealed in Panama Papers data leak in April 2016 by type [5]

The purpose of this paper is to enhance security awareness environment within an organization in order to overcome the various security threats, attacks and vulnerabilities through empowering modeling and simulation strategy based on network security framework models. It also meets the demands of the countermeasures strategy and policy of an organization.

The rest of the paper is structures as follows. Section II presents the cyber protect simulation tool. Section III presents security threats, attacks and vulnerabilities discussion. Section IV explains countermeasures strategy and policy. Section V discusses why an organization needs to adopt an appropriate network security framework model to enhance its network security environment. Section VI describes contribution of this paper by proposing a new concept proposal, called The Six-Ware Framework (The SWF). This contribution is a very early concept inspired by cyber protect simulation experiences. Section VII contains concluding remarks and future work for the SWF concept development.

II. WHAT IS CYBER PROTECT?

Cyber Protect is a network security simulation tool designed by the DISA [6]. It revolves around the purchase and application of information security countermeasures in a Local Area Network (LAN) environment. It takes place over four quarters. Each quarter the user makes decisions about what resources/countermeasures to purchase and put in place [7]. After making those decisions, the simulation is set in motion. The user is then subject to a variety of security attacks. The following cycle steps are repeated four times:

- First step, choose computer network security resources, e.g., user training, redundant systems, access control, virus protection, backup, disconnection, encryption, firewalls, and intrusion detection.
- Second step, applies/installs resources by drag and drop to a specific location on the cyber protect simulation dashboard.
- Third step, experiencing a variety of attacks. There are nine possible forms of attack, e.g., jamming, viruses, moles, social engineering, packet aniffers, data theft, data modification, flooding, and imitation (spoofing). The numbers and types of attacks are random; they can come from outside and inside an organization.
- Fourth step, receiving report indicating performance level. For every quarter the user receives a score sheet based upon how well they did in purchasing and applying resources to thwart the attacks.

In cyber protect simulation exercise, the user acts as an information leader within an organization. The user has full responsibility to protect or to defend his LAN department. Moreover, by utilizing cyber protect simulation dashboard, the user can freely setup the best and appropriate strategies of a LAN configurations which are expected to be immune from various types of threat, attack or data breach [8].

In order to successfully complete the simulation, meeting a "commanders" goal, the user needs to score 90 or above. As in real world situation, there is good and bad experiences and/or fortune associated with the simulation. A user might do very poorly in allocating his resources, yet through good fortune be subject to very few attacks, and therefore receive a final high score. At the other end of the spectrum, the user might do a pretty good job in allocating the resources, yet because of numerous attacks, the ending tally would not be good. Even with perfect "known" defenses, the enemy may still slip through.(see Figure 2).

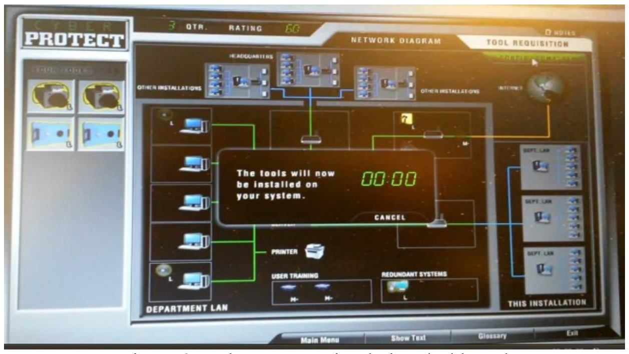


Figure 2. Cyber protect simulation dashboard

The objective of cyber protect simulation exercise is to produce a minimum 90% security readiness rating. If the user achieves this requirement, then the user can print out a certificate states that the user has passed network security readiness rating as an ultimate information leader in his organization [9]. For this simulation exercise, a remark of 94% out of 100% was obtained, which is an evidence that the network security design met the standards and passed successfully.

III. THREATS, ATTACKS AND VULNERABILITIES

During the process of cyber protect simulation exercise, the user will experience several types of threats, attacks and vulnerabilities e.g.,:

- Flooding, from Internet (external), where the symptom on incident report stating “Network server and/or Router function seriously impaired, degraded or crashed”.
- Viruses, from internal network stating “Network users report odd characters, noises, tunes, and/or messages appearing on work station screens. Network operations are unusual, degraded or crashed”.
- Packet Sniffer, from Headquarters (HQ) stating “Slight degradation in time required for network information transference”.
- Jamming, from HQ stating “Network Transmissions become unreliable or unreadable due to interfering signals”.
- Social engineering attack, from internal network stating “Report of suspicious attempts by outside individuals to gain access to information”.

To deal with those threats, attacks and vulnerabilities cyber protect simulation exercises was divided into four quarter tasks, each quarter consist of at least two threat types, attacks and vulnerabilities. Every result obtained in each quarter task is displayed into a form of quarter summary reports. Useful experiences during cyber protect simulation process whereby the user can investigate any failures in his network security at the previous quarter. The user determines why controls in place did not prevent threats, attacks and vulnerabilities, while making attempts to improve the network security system at the sub-sequent quarter.

IV. COUNTERMEASURES STRATEGY AND METHODOLOGY

Countermeasure strategy and methodology were needed during cyber protect simulation exercises. The user was asked to design a secure process, technology and personnel of the computer network systems, effectively and efficiently. The user can also identify residual risks of the modelled LAN. At this point, it was found that most of threats and attacks came from internal network; these are more difficult to tackle than the external ones (outsiders).

From the threat-driven approach perspective, most threats that came from insiders and outsiders (internet) can be handled effectively through a good methodology e.g., placing proper security and adequate peripherals, such as, firewalls, IDS and encryption, etc. The threat-driven approach is a methodology, a set of practices and a mindset. The primary purpose of this approach is to enable organizations to allocate the commensurate level of resources to defend their assets, to develop the inherent skills needed to support these efforts, and to align groups and teams into functional roles that will implement this approach [10]. Figure 3 shows a success countermeasure strategy for a LAN modeled configuration by developing appropiate security strategy, effectively and efficiently.

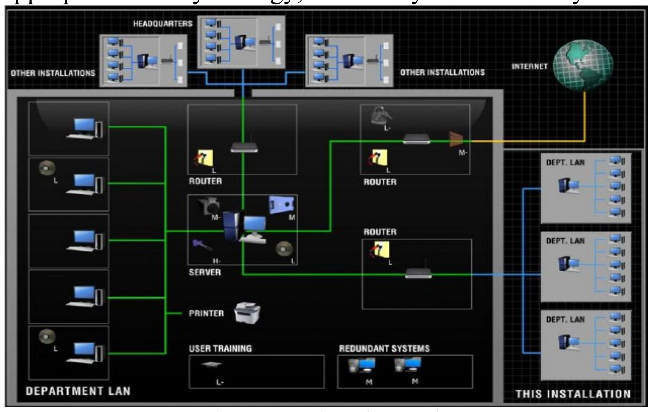


Figure 3. Design of secured LAN

It was found that the proper security strategy worked very well in the proposed modeled network security system; the strategy works as follows:

- First, configure one medium firewall and one low encryption system at the main router that is connected directly to the internet. The aim is to anticipate threats or attacks from outside the network, any kind of attacks from the Internet enter the network can be anticipated by a firewall system.
- Second, configure three low level of access control units at the entrance and exit of data communication lanes in the network to make it sure that there is no communication path that is not observed in the network. These access control units work as an early warning control system for the network administrator and it has the capability of monitoring all data transmission in the network.
- Third, complete system security in every servers with proper security equipments, e.g., high antivirus system, low level backup system, one medium Intruder Detection System (IDS) and one medium redundant system. This strategy can be applied to secure server from various attacks.
- Fourth, configure two low level backup systems on a particular client who has a high risk job in order to avoid from internal threats or breaches, especialy via social engineering attacks.

It was found that the proper implementation of countermeasure strategy is a crucial point in cyber protect simulation exercise. The countermeasure strategy might be implemented in various LAN departments, but it depends on its information security and risk management policies [11]. On the other hand, several countermeasure strategies, e.g., Defense-In-Depth Strategy by the US Homeland security or ProCurve-ProActive Defense Strategy by the Hewlett-Packard innovation center can be found on internet.

A. Defense-In-Depth Strategy

In October 2009, the US Homeland security developed a defense-in-depth strategy as a recommended practice in order to improve Industrial Control Systems (ICS) cyber security [12]. This strategy is not just about deploying specific technologies to counter certain risks, but it depends on how effective security program for an organization, its adherence and willingness to accept security as a constant constraint on all cyber activities.

Moreover, implementing an effective defense-in-depth strategy will require taking a holistic approach and leveraging all of an organization's resources in order to provide effective layers of protection. Figure 4 shows an overview on the key elements of a defense-in-depth strategic framework.

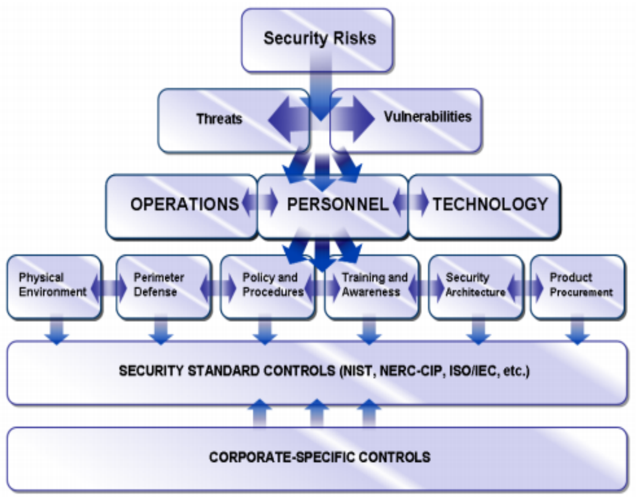


Figure 4. The strategic framework for cyber defense-in-depth

The basic principles of this framework are as follows:

- First, to know the security risks that an organization faces.
- Second, to quantify and qualify those risks.
- Third, to use key resources to mitigate security risks.
- Fourth, to define each resource's core competency and identify any overlapping areas.
- Fifth, to abide by existing or emerging security standards for specific controls.
- Sixth, to create and customize specific controls that are unique to an organization.

In order to implement a defense-in-depth strategy an organization will need to start at understanding its current risk. Risk for industrial control systems is best understood by knowing the threats and vulnerabilities that face an organization. The organization should undergo a rigorous risk assessment that covers all aspects to understand risk. Risk assessments are very crucial steps in defining, understanding, and planning remedial efforts against specific threats and vulnerabilities. All level areas and levels in the organization, including executives, must support the valuable risk assessments which are constantly updated at timely intervals.

B. ProCurve-ProActive Defense Strategy

In February 2007, the Hewlett-Packard (HP) innovation proposed a new alternative for network security: a comprehensive security vision and strategy that arises directly from the revolutionary ProCurve Adaptive EDGE Architecture™ (AEA). This defense strategy embraces distributed intelligence at the network edge and takes a holistic approach to an organization's or company's networking. The HP innovation declared a new security vision, called ProCurve-ProActive Defense strategy. This was claimed as the first approach that combined proactive security offense techniques with steadfast traditional defense security techniques simultaneously, at the edge of the network, where users connect.

As such, ProCurve-ProActive defense strategy is expected to change dramatically how network security is deployed from now on. ProCurve-ProActive defense strategy delivers a trusted network infrastructure that is immune to threats, controllable for appropriate use and able to protect data and integrity for all users. The three main pillars of the ProActive Defense strategy are as follows:

- Access Control, proactively prevents security breaches by controlling which users have access to systems and how they connect in a wired and wireless network.
- Network Immunity, detects and responds to internal network threats such as virus and worm attacks; monitors behavior and applies security information intelligence to assist network administrators maintain a high level of network availability.
- Secure Infrastructure, secures the network for policy automation from unauthorized extension or attacks to the control plane; includes protection of network components and prevention of unauthorized managers from overriding mandated security provisions; also includes privacy measures to ensure the integrity and confidentiality of sensitive data: protection from data manipulation, prevention of data eavesdropping, end-to-end VPN support for remote access or site-to-site privacy, and wireless data privacy (see Figure 5).

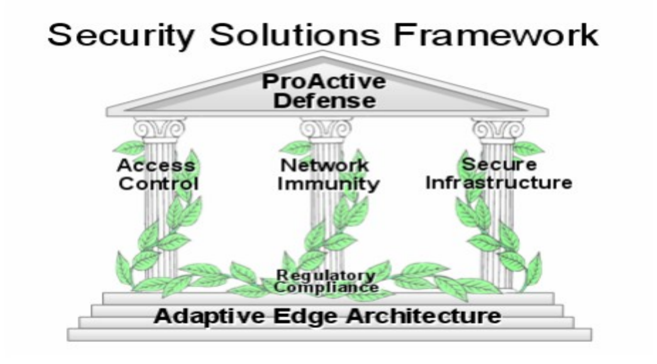


Figure 5. The Three Pillars of Access Control, Network Immunity and Secure Infrastructure

One of unique aspects of the ProCurve-ProActive defense vision and strategy is that it combines both the security offense and security defense at the same time and, most importantly, at the network edge [13]. This combined offense and defense is possible only because ProActive defense is based on AEA principles, which drive intelligence to the network edge while retaining centralized control and management. ProActive defense strategy includes characteristics such as the following:

- Additional enhancements to Identity Driven Manager, such as clientless and agent-based endpoint integrity with flexible remediation and a vulnerability assessment framework.
- Additional enhancements to Network Immunity Manager, such as increased network behavior anomaly detection (NBAD) capabilities.
- Enhanced policy control at the edge, including Web-Auth with clientless endpoint integrity authentication.
- Standards-based endpoint integrity, with trusted agent access for LANs, WANs and WLANs.

V. NETWORK SECURITY FRAMEWORK MODELS

Based on cyber protect simulation experience, organizations need to adopt an appropriate security policy as well as planning and deployment in order to enhance its network security. Every personnel within the organization, from senior level management down to the staff level, must be fully aware of the importance of enterprise information security. All employees should understand the underlying significance of security policy, planning and deployment of the organization. There are several models providing security framework or security reference model, available in the market, namely the US National Institute of Standards and Technology (NIST) or the Control Objectives for Information and related Technology (CobiT) security framework, etc.

A. *The NIST cyber security framework*

In February 2013, the US President issued an Executive Order (EO) 13636, in order to improving national critical infrastructure cybersecurity. The EO states: "It is the policy of the United States to enhance the security and resilience of the Nation's critical infrastructure and to maintain a cybersecurity environment that encourages efficiency, innovation and economic prosperity while promoting safety, security, business confidence, privacy and civil liberties."

This order directed NIST to work with stakeholders to develop a voluntary framework – based on existing standards, guidelines, and practices - for reducing cyber risks to critical infrastructure (NIST 2014) [14]. NIST framework consists of standards, guidelines, and practices to promote the protection of critical infrastructure. NIST framework is organized into five basic cybersecurity activities:

- Identify (to develop the organization's understanding to manage cybersecurity risk to systems, assets, data and capabilities).
- Protect (to develop and implement the appropriate safeguards to ensure delivery of critical infrastructure services).
- Detect (to develop and implement the appropriate activities to identify the occurrence of cybersecurity events).
- Respond (to develop and implement the appropriate activities to take action regarding a detected cybersecurity event).
- Recover (to develop and implement the appropriate activities to maintain plans for resilience and to restore capabilities or services that were impaired due to a cybersecurity event).

Each of the functions are then divided into categories to define more specific security practices and capabilities (e.g., asset management, access control). Subcategories describe more detailed or technical controls needed to meet objectives within each category (see Table I).

TABLE I. THE NIST CYBER SECURITY FRAMEWORK

Func-tions	Categories	Sub-categories	Information References
Identify	• Asset Management • Governance	• Invetory devices, systems and software, etc.	• NIST 800-53 CM-8, CA-2, etc.
Protect	• Access Control, etc.	• Review access periodically • Two-factor authentication	• ISO 27001 A6, A9, A11, A13, etc.
Detect	• Detect & Monitor for anomalies and events	• Review logs for suspicious activity, etc.	• NIST 800-53 AU-6, CA-7, etc.
Respond	• Mitigation of security events, etc.	• Report suspicious events, etc.	• ISO 27001 A6, A16, etc.
Recover	• Recovery planning, improve-ments and communi-cation	• Recovery plan • Manage public relations • Repair reputation	• NIST 800-53 CP-10, IR-4, IR-8, etc. • ISO 27001 A16, etc.

B. *The CobiT security framework*

CobiT is an Information Technology (IT) governance framework developed by the Information System Audit and Control Association (ISACA). CobiT consists of acquire and maintain application software; acquire and maintain technology infrastructure; develop and maintain procedures, install and accredit systems and manage changes. In April 2012, CobiT 5 was released, with the concept of enterprise governance of IT as a foundation.

CobiT 5 provides a comprehensive framework that assists enterprises to achieve their objectives for the governance and management of enterprise IT [15]. CobiT 5 brings together the five principles that allow the enterprise to build an effective governance and management framework based on a holistic set of seven enablers that optimises information and technology investment and use for the benefit of stakeholders (see Figure 6).

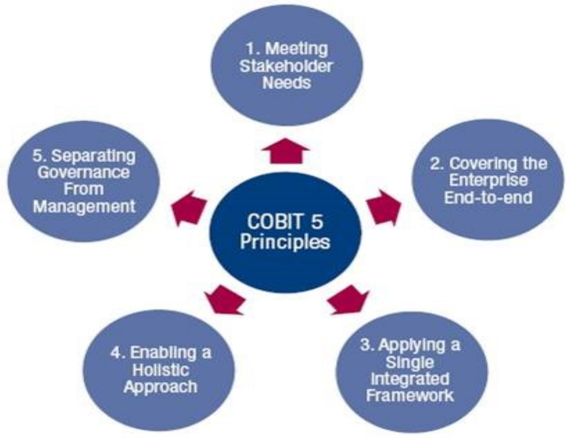

1. Meeting Stakeholder Needs
2. Covering the Enterprise End-to-end
3. Applying a Single Integrated Framework
4. Enabling a Holistic Approach
5. Separating Governance From Management
COBIT 5 Principles

Figure 6. The strategic framework for cyber defense-in-depth

- Principle 1, meeting stakeholder needs - stakeholder needs are translated into specific enterprise, IT-related goals and enabler goals.
- Principle 2, covering the enterprise end-to-end – governance and management of information and related technology is addressed from an enterprise-wide, end-to-end perspective.
- Principle 3, applying a single integrated framework - COBIT 5 defines the overarching governance and

management framework that has been designed to integrate seamlessly with other good practice guidance, e.g., ISO 38500.
- Principle 4, enabling a holistic approach – the seven categories of inter-connected enterprise enablers are set out below (see Figure 7).
- Principle 5, separating governance from management. CobiT 5 advocates that organisations implement the key governance and management processes (see Figure 8).

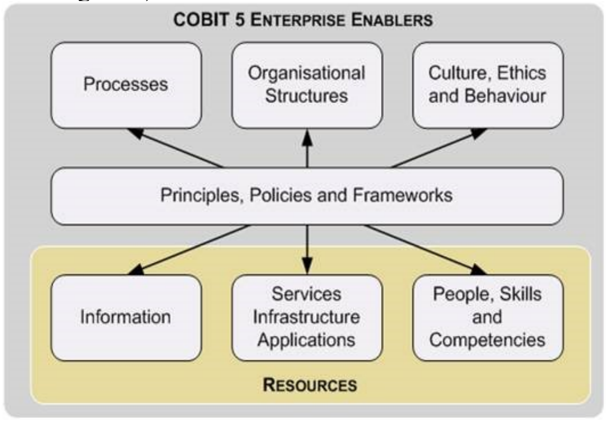

COBIT 5 ENTERPRISE ENABLERS
Processes
Organisational Structures
Culture, Ethics and Behaviour
Principles, Policies and Frameworks
Information
Services Infrastructure Applications
People, Skills and Competencies
RESOURCES

Figure 7. The seven categories of CobiT 5 enterprise enablers

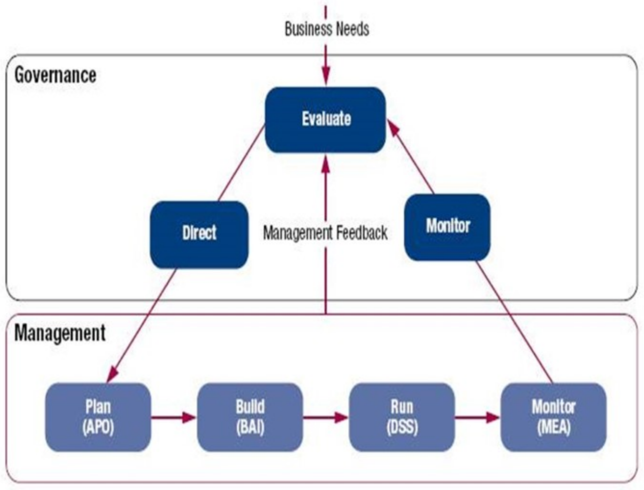


Figure 8. Separating Governance from Management

VI. CONTRIBUTION

This paper contributes an initial security framework concept, so called, The Six-Ware Framework (The SWF). The SWF concept is a comprehensive security solution to enhance an organization's network security resilience from various threats, attacks and vulnerabilities. This is an operational-level security strategy that enables to figure out the most efficient and effective actions that may lead to the success of cyber security operation [16]. The idea behind this new concept was inspired by NIST cyber security platform version 1.0., dated 12 February 2014. The SWF concept tries to elaborate NIST cyber security framework to be more practical for the operational level. The security framework disscussion can be found also in mashup web data extraction system [18].

The SWF concept contributes a common thought to understanding, managing, and expressing network security risks, both internally and externally. The SWF concept contributes increased security awareness environment within an organization where it requires internal/external risk assessment and also threat analysis policies. All levels employees in the organization, ranging from highest level to lowest level must be actively involved in the SWF concept implementation. Otherwise, they can not obtain better understanding of how threats or attacks can be carried out successfully across the entire organization.

A. The SWF enablers

The SWF enablers provide a set of activities, which consists of six main variables, sub-variables, indicators and information references (e.g., reference guidances). The SWF enablers are not only a set of checklist of actions to perform, but it presents key network security solutions to manage security risk and analysis in an organization computer network [19]. The SWF enablers comprises six main aspects, e.g., Brainware, Hardware, Software, Infrastructureware, Firmware, Budgetware (see Table II).

- Brainware or human factor, is the main aspect in network security environment. This variable becomes top list variable within the SWF concept. From network security perspective, it commonly known that human is the weakest link in information security environment. Human factor plays dominant role to enhance or on the contrary, to disrupt all efforts of existing information security wityhin an organization. Therefore, organizations must have function or position related to information security, e.g., Chief Information Security Officer (CISO). The CISO is a company's top executive who is responsible for security of personnel, physical assets, data and information in both physical and digital form. The CISO position has increased in the era of cyberspace where it becomes easier to steal sensitive company information. One of CISO's responsibilities is to conduct information security certification programs to all level employees. The intention is to produce "information security awareness employees" related to their position and function.
- Hardware, plays dominant role in handling threats, attacks and vulnerabilities. CISO has to teach all level employees how to use and treat organization's hardware devices safely and wisely. It is because a high-level hacker is not just relying on a specific technique, but still combined with the conventional attack, e.g., social engineering attack. Combination of internal risk assessment and threat analysis are extremely needed, e.g., controlling individual access into the organization's premises or facilities, locking systems and removing unnecessary CD-ROM or

USB thumb drives, or monitoring and protecting the security perimeter of organization's facilities, etc.

- Software, relates to utilization of software applications security which are used daily in the office, e.g., email, website, social media and other applications. High security awareness is really required because a high profile attacker will always kept on trying to infect or inject malicious emails and its attachments or invite to visit malware-infected websites. The attackers are also constantly introducing new threats although various cyber security application tools are available in the market.
- Infrastructureware, has an important role in facilitating secure organization network infrastructure, e.g., monitoring network from various threats, attacks and vulnerabilities. Nowadays, most of organizations have been highly dependent on Internet access. On the other hand, not all of employees have a good level understanding about security risks they might face in the office, where this condition is making the organization's network infrastructure more vulnerable.
- Firmware, includes documentation of an organization security strategy and policy, standard operating procedures (SOPs), business continuity plans (BCPs), network security frameworks or international security standardizations compliance to International Organization for Standardization (ISO), such as ISO 27001:2013 [18]; NIST cyber security framework version 1.0 or government security policy and strategy [20], etc.
- Budgetware, plays important and strategic role in facilitating implementation of the five-ware variables above. It is because an organization is urged to provide big enough money or sufficient budget to purchase e.g., network security application tools, patching systems, software licenses, training and education, certification programs, etc. It is highly recommended top level management must put this matter as a high level priority in order to build information security awareness. Allocating sufficient information security budget could protect the entire network system. Otherwise, they will face organization's significant financial losses, etc.

TABLE II. THE SWF CONCEPT (ENABLERS AND COMPONENTS)

Aspe-cts	Varia-bles	Sub-varia-bles	Indicators	Infosec References
Brain ware	• CISO, etc.	• Security training, etc.	• Security Aware-ness	• CISSP, CISA, etc.
Hard ware	• Server Farms	• USB, etc.	• No com-promises	• Bench mar-king, etc.
Soft ware	• Applic ation	• MS Office, etc.	• No pira-ted Appl. etc.	• Regular updates,etc
Infra struc-ture-ware	• Net-work Infra-struc-ture	• Firewalls. • IDS. • DMZ, etc.	• No net-work security breaches, etc.	• Self penetration testing, etc.
Firm ware	• Secu-rity hand book	• Bussiness Continui-ty Plan	• Good Bussi-ness pro-cesses	• NIST. • ISO 27001, etc.
Bud-get ware	• Suffi-cient budget	• Buy soft-ware licen ses, etc.	• Licences always updated, etc.	• Allocated budget policy, etc.

B. The SWF component

The SWF component works together as follows:

- Variables, organize network security fundamental aspects as enablers, e.g., brainware, hardware, software, infrastructureware, firmware and budgetware) at highest level. These variables help an organization in managing its security risk and analysis by organizing or clustering information, threats and attacks activity. Variables align with security and policy framework to reduced impact to organization quality of services (QoS) e.g., investments in human resources, planning and budgeting exercises or recovery actions, etc.
- Sub-variables, are sub-divisions of a variable closely tied to a particular (for example, brainware variable) security awareness activities e.g., "security awareness", "socialization and training", "cyber security certification program", etc.
- Indicators, are sub-divisions of a sub-variable, divided into technical outcomes. Indicators provide a set of results to achieve outcomes for each sub-variable. Indicators example (like security awareness sub-variable) e.g., "conducting security awareness training program"; "socializing and implementing security awareness culture in the company"; or "notifications from any social engineering attacks or security breaches that are being investigated", etc.
- Information References (IR), consists of network security standards, guidelines, methods and practices to achieve solutions or outcomes associated with each indicator. IR which presented in the SWF concept are illustrative and not complete. Examples of IR (like conducting security awareness training program indicator) e.g., "certified ethical hacking (CEH) course from EC-council"; "DoD information assurance awareness training"; and "Achieving ISO 27001 Certification"; etc.

The SWF component provides a set of activities to achieve specific network security outcomes, and references examples of guidance to achieve those outcomes. The SWF component is not a checklist of actions to perform. It presents key cybersecurity outcomes identified by organization as helpful in managing the risk within organization network security environment.

VII. Conclusion and Future Work

Cyber protect simulation is a good simulation tool, but it needs to be developed to face the growth of new variants of security threats, attacks and vulnerabilities. It provides users with useful experiences of tactical and strategical security situation awareness. The users are given the freedom to model and simulate the best strategy to defense his secured LAN configurations efficiently and effectively.

In this paper, the SWF concept can not be compared with the NIST, because it is just an initial proposal to enhance an organization's network security environment. In the future, the SWF concept needs to be developed more in-depth through further research on specific areas, e.g., determining more technically and specifically security framework variables, sub-variables, indicators and information references. The SWF concept acts as an early warning system measurement within an organization. It portraits the existing LAN security environment while finding the root cause of security loopholes. It can be concluded that to achieve a totally secure network environment is very difficult.

Acknowledgment

The authors would like to give high appreciation to the i-College, IRMC, NDU, Washington, DC., USA., for giving a valuable chance to attend cyber security for information leaders course in March 2015. The authors would like also to thank the Lemhannas RI and the Nuffic (Project Niche IDN 143) for its financial support so the authors can attend and submit this academic paper in the ICIMP 2016 International Conference, Valencia, Spain, May 22 to 26, 2016.

References

[1] J. H. Saunders, "The Case for Modeling and Simulation of Information Security," National Defense University. http://www.johnsaunders.com/papers/securitysimulation.htm, last accessed April 2016.

[2] Sara Peters, "7 Lessons From The Panama Papers Leak," vulnerabilities/ threats, http://www.darkreading.com/vulnerabilities---threats/7-lessons-from-the-panama-papers-leak/d/d-id/1324976, last accessed April 2016.

[3] Swati Khandelwal, The Panama papers-Biggest leaks in History Exposes Global Corruption, The Hacker News, http://thehackernews.com/2016/04/panama-paper-corruption. html, April 3, 2016.

[4] Statista.com, "Number of files revealed in Panama Papers data leak in April 2016, by type," http://www.statista.com/statistics/531286/ panama-papers-data-type/, last accessed May 2016.

[5] Statista.com, "Number of files revealed in Panama Papers data leak in April 2016 by type", http://www.statista.com, last access April 2015.

[6] The i-college, Cyber Cecurity for Information Leaders course, "Cyber Protect Simulation Exercises," National Defense University (NDU), Washington, DC., USA, March 2015.

[7] Vicente Pastor, Gabriel Díaz and Manuel Castro, "State-of-the-art Simulation Systems for Information Security Education, Training and Awareness," IEEE EDUCON Education Engineering 2010, The Future of Global Learning Engineering Education, 978-1-4244-6571-2/10, April 14-16, 2010, Madrid, Spain.

[8] Cyber Protect Network Defense Simulation Tool, https://ndu.blackboard.com and http://iatraining.disa.mil/eta/ cyber-protect/launchpage.htm, the i-college, NDU, Washington, DC, USA, March 2015.

[9] Ann O'Brien, "Effective Learning Strategies: Cyber Protect – Learning About System Security", Wisconsin School of Bussiness, adapted from Jim Mensching, Chicago State University, USA.

[10] Michael Muckin, Scott C. Fitch, "A Threat-Driven Approach to Cyber Security: Methodologies, Practices and Tools to Enable a Functionally Integrated Cyber Security Organization," Lockheed Martin Corporation, http://lockheedmartin.com/content/dam/ lockheed/data/isgs/documents/Threat-Driven%20Approach%20 whitepaper.pdf, last accessed May 2016.

[11] Cyber Security for Information Leaders course, "Information Security and Risk Management," CISSP Textbook Reading, Chapter 3, the i-college, NDU, Washington, DC, USA, March 2015.

[12] The US Homeland Security, Recommended Practice: Improving Industrial Control Systems Cybersecurity with Defense-In-Depth Strategies,https://ics-cert.us-cert.gov/sites/ default/files/recommended practices/Defense_in_Depth_Oct09.pdf, October 2009, last accessed May 2016.

[13] The Hewlett-Packard (HP) innovation, "ProCurve-ProActive Defense: A Comprehensive Network Security Strategy," Pro Curve Networking, February 2007, http://www.hp.com/rnd/pdfs/ProCurve _Security_paper_022107.pdf, last accessed May 2016.

[14] The National Institute of Standards and Technology (NIST), "Framework for Improving Critical Infrastructure Cybersecurity Version 1.0.," http://www.nist.gov/cyberframework/upload/cyber security-framework-021214-final.pdf, February 12, 2014, last accessed May 2016.

[15] ISACA,"COBIT 5 Framework: A Business Framework for the Governance and Management of Enterprise IT," http://www.isaca. org/cobit/pages/cobit-5-framework-product-page.aspx, last accessed April 2016.

[16] Chen, J., and Duvall, G., "On Operational-Level Cybersecurity Strategy Formation," Journal of Information Warfare: 13.3: 79-87. SSN 1445-3312 print/ISSN 1445-3347 online, 2014.

[17] Rudy AG Gultom, "Proposing the new Algorithm and Technique Development for Integrating Web Table Extraction and Building a Mashup," Journal of Computer science, Science Publication, NY, USA, DOI: 10.3844/jcssp.2011.129.142, http://www.thescipub.com/ issue-jcs/7/2, 25 February 2011.

[18] Rudy AG Gultom, "The Six-Ware Framework Proposal: A New Comprehensive Cyber Security Framework To Defend Your Network From Social Engineering Attack," Final Paper, i-college, IRMC, National Defense University, Washington, DC., USA, 19 March 2015.

[19] ISO, "ISO/IEC 27001: 2013, Information Technology-Security Techniques-Information Security Management Systems-Requirements,"http://www.iso.org/iso/catalogue_detail?csnumber=54 534, last accessed April 2016.

[20] Adam Quinn, "Obama's National Security Strategy Predicting US Policy in the Context of Changing Worldviews,"US Research Papaer, Project 2015, https://www.chathamhouse.org/sites/files/ chathamhouse/field/field_document/20150109ObamaNationalSecurit yQuinn.pdf, last accessed April 2016.