ICIMP 2019

The Fourteenth International Conference on Internet Monitoring and Protection

ISBN: 978-1-61208-729-0

July 28 – August 2, 2019

Nice, France

ICIMP 2019 Editors

Petre Dini, IARIA, USA

ICIMP 2019

Forward

The Fourteenth International Conference on Internet Monitoring and Protection (ICIMP 2019), held between July 28, 2019 and August 02, 2019 in Nice, France, continued a series of special events targeting security, performance, vulnerabilities in Internet, as well as disaster prevention and recovery. Dedicated events focused on measurement, monitoring and lessons learnt in protecting the user.

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.

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.

The conference included academic, research, and industrial contributions. It had the following tracks:

- Internet traffic surveillance and interception
- Internet monitoring and protection

We take here the opportunity to warmly thank all the members of the ICIMP 2019 technical program committee, as well as all 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 effort to contribute to ICIMP 2019. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

We also thank the members of the ICIMP 2019 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope that ICIMP 2019 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the area of Internet monitoring and protection. We also hope that Nice, France provided a pleasant environment during the conference and everyone saved some time to enjoy the charm of the city.

ICIMP 2019 Chairs

ICIMP Steering Committee

Sathiamoorthy Manoharan, University of Auckland, New Zealand
Terje Jensen, Telenor, Norway
Christian Callegari, University of Pisa, Italy

ICIMP Industry/Research Advisory Committee

Daisuke Mashima, Advanced Digital Sciences Center, Singapore
Bernhard Tellenbach, Zurich University of Applied Sciences, Switzerland
Miroslav Velev, Aries Design Automation, USA
Pethuru Raj, Reliance Jio Infocomm. Ltd. (RJIL), India

ICIMP 2019

Committee

ICIMP Steering Committee

Sathiamoorthy Manoharan, University of Auckland, New Zealand
Terje Jensen, Telenor, Norway
Christian Callegari, University of Pisa, Italy

ICIMP Industry/Research Advisory Committee

Daisuke Mashima, Advanced Digital Sciences Center, Singapore
Bernhard Tellenbach, Zurich University of Applied Sciences, Switzerland
Miroslav Velev, Aries Design Automation, USA
Pethuru Raj, Reliance Jio Infocomm. Ltd. (RJIL), India

ICIMP 2019 Technical Program Committee

Andrea F. Abate, University of Salerno, Italy
Irfan Ahmed, University of New Orleans, USA
Hasan Ibne Akram, Matrickz GmbH, Germany
Lasse Berntzen, University of South-Eastern Norway, Norway
Abdelmadjid Bouabdallah, Université de Technologie de Compiègne, France
Aymen Boudguiga, Institute for Technological Research SystemX, France
Jean-Louis Boulanger, CERTIFER, France
Christian Callegari, University of Pisa, Italy
Paolina Centonze, Iona College, USA
Yeh-Ching Chung, Tsinghua University in Shenzhen, China
Domenico Ciuonzo, University of Naples Federico II, Italy
Paolo D'Arco, University of Salerno, Italy
Rubens de Souza Matos Júnior, Federal Institute of Education, Science, and Technology of Sergipe, Brazil
Lorenzo DeCarli, Worcester Polytechnic Institute (WPI), USA
Hervé Debar, Télécom SudParis, France
Raffaele Della Corte, "Federico II" University of Naples, Italy
Tadashi Dohi, Hiroshima University, Japan
Serena Doria, University G.d'Annunzio Chieti-Pescara, Italy
Christian Esposito, University of Salerno, Italy
Eduard Marin Fabegras, KU Leuven, Belgium
Parvez Faruki, Malaviya National Institute of Technology, India
Eduardo B. Fernandez, Florida Atlantic University, USA
Pietro Ferrara, JuliaSoft, Italy
Arpad Gellert, Lucian Blaga University of Sibiu, Romania
Kambiz Ghazinour, Kent State University, USA
Dimitra Georgiou, University of Piraeus, Greece

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Framework for Creating Security Functions Based on Software Defined Network

Dobrin Dobrev
Technical University Sofia
Sofia, Bulgaria
e-mail: d_dobrev@tu-sofia.bg

Dimiter Avresky
IRIANC
Munich, Germany
e-mail: autonomic@irianc.com

***Abstract*—In this work, we propose a framework for security based on Virtual Security Functions, OpenFlow, Software Define Networks (SDN), Mininet, Pox Controller and Virtual Switches. By using the OpenFlow protocol in the virtualized environment of SDN, we are capable of analyzing the data streams in the network environment. An SDN controller, staying on top of the entire infrastructure, is capable of orchestrating network segment(s). By creating different virtual security functions, we have the possibility to increase network security and to avoid loops. In this paper, the process of loop elimination is achieved by automatically reconfiguring the security function by creating a spanning tree. By using the scalability of the virtual controller, we can simplify the network administration. The main benefit is that different systems like switches, firewalls, and Intrusion Detection Systems (IDS) will be replaced by the controller with Virtual Security Functions (VSF). The target is to increase availability by presenting functions that avoid loops in the network. VSF will allow to exploit the framework in order to eliminate different attacks, such as congestion driven attacks, Distributed Denial-of-Service (DDoS), Media Access Control (MAC) address spoofing, man in the middle attack and Synchronize (SYN) flood attacks. All functions can be run in parallel and we can increase the availability of the system.**

Keywords-security function; network function virtualization; virtualization, openflow; flowtable; controller.

I. INTRODUCTION

The classic model for information security defines three objectives of security [1]: confidentiality, integrity, and availability. The main goal of availability is to ensure that information and resources are available to those who need them i.e. security will be guaranteed. One of the possible ways to attack the network infrastructure is over a congested link. It is widely known that congestions can generate network loops and different types of attacks over the Internet. The loops can be generated when there is more than one path between two hosts. Attacks can be created by means of loops in the networks. Problems will appear via attacks that disrupt the regular communications. In this paper, the process of loop elimination is achieved by automatically reconfiguring the Spanning Tree Protocol (STP). We use STP in situations where we want redundant links, without loops. This process is automatic and it avoids deadlocks in our topologies. The proposed framework provides scalable solutions under the SDN [5] environment. Mininet [27] is used as an emulated environment. It is combined with OpenFlow [6].

II. RELATED WORKS

The major work in this paper is related to SDN. The novelty of our work is in the approach of targeting the spanning tree in order to ensure availability in the SDN environment. Similar research targeting availability can be found under research paper [2]. In this paper, the authors are creating different paths with specific metrics via SPT. Spanning tree has been targeted as a security issue as well in research paper [32]. In this work, by using POX controller and SPT, the authors are analyzing the network threats.

Based on our framework, different security functions can be created in order to ensure a higher level of security. In addition to loop elimination, by using this framework, the developers can create security functions to handle Distributed Denial of Service (DDoS) [4] and Intrusion Detection Systems (IDS) [31].

III. TECHNOLOGIES USED

A. *Software-defined networking layered segmentation*

SDN is comprised of multiple kinds of network technologies designed to make the network more flexible and agile [5]. It is an approach for using open protocols, such as OpenFlow [6]. The decision on how packets must be transferred is taken by the SDN Controller. With this type of technology, it is possible to change the topology without touching individual switches [7]. The delivery is possible with the network virtualization and the separation of data plane and control plane [8]. The Application layer and the Control layer [10] together can be considered as a Logical layer on top of the Infrastructure layer, which is the physical part of the communication [10]. At the Application layer, we are creating the Loop Elimination Function as a Security function to avoid loops by using a spanning tree, as shown on Figure 1.

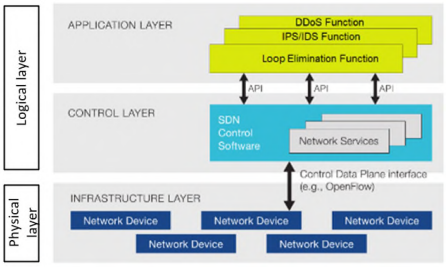

Figure 1. SDN Layers and Security function

 ISBN: 978-1-61208-729-0

The Infrastructure layer is responsible for data transferring. SDN Virtual Switches (*Vswitch*) are used in the SDN environment. *Vswitches* are supporting protocols such as NetFlow, sFlow and OpenFlow. They are utilised to connect the Infrastruture layer with the upper Control Layer via Control Data Plane Interface, as shown in Figure 1. The Control Layer is the middle layer, where network services are located. This layer maintains a global view of the network and provides hardware abstractions to SDN applications. The implementation of the SDN control plane can follow a centralized, hierarchical, or decentralized design [10]. This is a logical entity that receives instructions or requirements from the SDN Application layer and relays them to the Infrastructure layer. There are a lot of different options for applications on this level, such as: Intrustion Prevention System (IPS), Distributed Denial of Service functions protecting SYN Flood attaks, ping of death, amplification attack, and many more. Other services can be added as well, such as networking management, analytics, or business applications. SDN gives the flexibility to every programmer and administrator to choose the best path for their services by configuring the SDN controller with a custom policy. This way, they are able to prioritize, deprioritize, or even block specific types of packets with a very granular level of control [12]. Also, it can be used in a cloud computing, multi-tenant architecture, because it gives the traffic the possibility to be managed in more efficient and flexible manner [13]. Using this model, the owner of the data has the control of the data. Several research ideas based on SDN/OpenFlow have been proposed in [14], since the publication of DDoS protection for SDN [15], which is a key component in realizing the SDN concept for decentralized protection. We will focus on the security protection realized by VSF. Open Virtual Network (OVN) [3] is an example of an open source product using OpenFlow. According to OVN, some community main challenges are: Congested network link between physical switches; Poorly performing virtual switches; Overloaded servers and Distributed Denial of Service (DDoS) attacks. OVN are using their proprietary model [3] for congestion elimination. The difference is that we are using a spanning tree and *Mininet* environment for avoiding loops. According to the research in [32], STP is used for threat analyses of network. In our research, we are focusing more on the aspect of availability assured after the elimination of loops.

B. OpenFlow as a technology

OpenFlow is a communications protocol that gives access to the forwarding plane of a switch or router over the network. This is the protocol used to connect the Infrastructure layer and the Control layer. With OpenFlow, packet-moving decisions are centralized. Having the control separated from the forwarding allows for a more sophisticated management. This type of protocol allows a more effective use of the network resources compared to the traditional networks [16]. OpenFlow is layered on top of the Transmission Control Protocol (TCP). The Datapath of an OpenFlow switch is determined via Flow Tables. A Flow Table matches incoming packets to a particular flow and specifies the functions that are to be performed on the packets. Flow Tables entrees can be divided in three sections: rules, actions, and states. Packets are assigned actions based on the rules that are matched to the Flow Table [17]. Those actions can be: to forward the packet, to send it to the controller, to send it to a normal processing pipeline, or to even perform NAT by changing the source IP address. The controller will manage all communications and will have visibility on all types of protocols. On the lower level, the controller has the knowledge of switch ports, source and destination IPs, of Media Access Control (MAC) addresses and of Layer 4 ports.

C. SDN security functions

SDN is a new paradigm and technology that is currently widely developed. After the development of OpenFlow, the separation of data plane and control plane is now possible, although at the same time there is still an issue with the security. This area needs to be developed further in order to ensure a proper level of protection. OpenFlow provides critical information for any connection. It can be pointed out that, by using a Flow Table, every connection can be analyzed by the controller and, based on this analysis, traffic can be allowed or rejected. In order to ensure better security, a lot of security functions are under development currently. There are theoretical solutions for firewalls, IPS/IDS, stateful firewalls; however, currently, there is no solid solution for switch fails redundancy. [2] discusses that, when a switch fails, its immediate downstream switch(es) will take time to recalculate paths and to assure availability. A downstream switch is protected if it has a neighbor whose path to the controller is not affected by this failure. By rerouting its traffic to this neighbor, the protected switch will bypass the failure. Nowadays, enhancing the security of SDN, improving autonomy and ensuring privacy of the data can be improved significantly. This could be achieved by creating virtual security functions with a different scope. In order to implement additional level of security in SDN, it has to be applied on the Application layer. To ensure a sufficient level of security according to ISO 27001[20] in a traditional environment, different hardware devices need to be used. With SDN, additional functions can be developed, such as Congestion elimination, Denial of Service (DoS), Distributed Denial of Service (DDoS), Intrusion Prevention System (IPS), Intrusion Detection System (IDS), Slower attack on Level 7, or layer 2 attacks using MAC address spoofing techniques. The main functions under development are firewalls [21]. They are using OpenTables in order to assure Datapath and to provide static control on incoming and outgoing traffic. After the development, Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) [22] have been added as functionalities on SDN firewall. Snort IDS [23] is an example of already developed IDS with predefined signatures. Applying/Using the SDN model allows those virtual security functions to operate on the Application layer and VSF are deployed over a virtualized environment.

 ISBN: 978-1-61208-729-0

IV. PROPOSAL

The area we are working on now will be situated at the Application layer. Security functions for each type of attacks can be implemented at this layer. Problems such as compatibility are eliminated because the entire environment is virtualized. Virtualized Network Functions (VNF) is an emerging paradigm that builds on cloud computing and the virtualization technology to eliminate the drawbacks of traditional physical network infrastructure. Different types of SDN applications can be created as a security function. Using Network Functions Virtualization Infrastructure (NFVI) allows all hardware elements to be located on a physical layer and all services to be deployed on a virtualized environment, as shown on Figure 2.

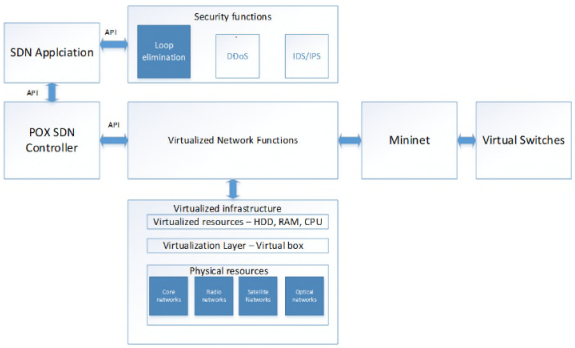

Figure 2. Virtual framwork with SDN properties

VNF is used for building blocks. They are performing the same services as the traditional firewalls or IDS/IPS, but on a virtualized environment. Loop elimination is the function that we focus on in our research.

V. SDN ENVIRONMENT

Our development environment is entirely virtual, as we are using a Virtual box as hypervisor. Oracle VM VirtualBox is a free and open-source hypervisor for x64bit computers currently being developed by the Oracle Corporation [25]. We have installed on this hypervisor one Linux with Ubuntu 14.04.4 and a customized version on Linux for Mininet 2.2.2. We used the Ubuntu Virtual machine to install the POX Controller [26] and we used Mininet to emulate the SDN environment. In this environment, one SDN switch and two emulated hosts are utilized [27]. In order to be connected to the host environment, we are using Putty [28], a terminal emulator, and Xming [29] X11 display server.

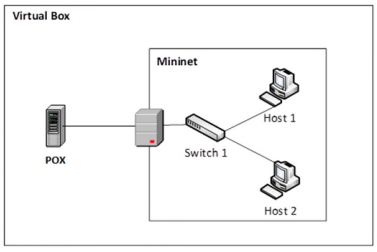

Figure 3. Virtual framwork with SDN properties

We are creating different virtual security functions for mitigating several types of attacks by using a virtualized environment. Examples of security functions are Loop Elimination, DDoS and IDS/IPS, as shown on Figure 3. In this paper, we are concentrating mainly on the Loop Elimination security function by using spanning tree.

VI. EXPERIMENTAL RESULTS

A. *Delay analyses in OpenFlow infrastructure topology*

By using the Mininet environment, different types of topologies have been created. We can use those types of graphs in order to describe our framework`s environment and to test our concepts and models. The presented framework creates different types of topologies and ensures communications between all hosts. One example of topology is shown in Figure 4.

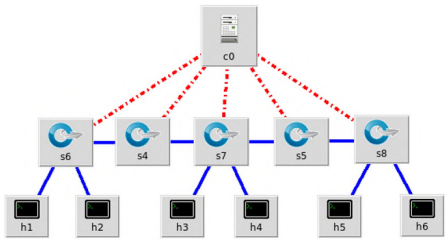

Figure 4. OpenFlow infrastructure topology

During this deployment, we used POX as controller and we were able to orchestrate the entire communication. Based on the virtualized environment, we established connections between each host orchestrated by an SDN controller and by using OpenFlow as a communication protocol, console output, as shown in Figure 5.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h4 h5 h6 h2 h3
h4 -> h1 h5 h6 h2 h3
h5 -> h1 h4 h6 h2 h3
h6 -> h1 h4 h5 h2 h3
h2 -> h1 h4 h5 h6 h3
h3 -> h1 h4 h5 h6 h2
*** Results: 0% dropped (30/30 received)
```

Figure 5. Host reachability by using Mininet and OpenFlow

We use ICMP as a protocol for demonstration purposes. Based on the results, we were able to analyse and to calculate different communication delays. It can be easily seen that the first packet needs more time to arrive at the destination than the following packets and we have zero percent packet lost. This is represented in Table I.

TABLE I. PING DISTRIBUTION

	Table with times in milliseconds					
	h1	***h2***	***h3***	***h4***	***h5***	***h6***
h1 first ping	x	17.30	24.10	37.81	76.30	13.70
h1 second ping	x	0.36	0.78	0.63	1.00	2.58
h2 first ping	14.50	x	11.00	47.50	70.60	61.41
h2 second ping	0.46	x	0.63	0.65	0.98	1.35

 ISBN: 978-1-61208-729-0

h3 first ping	55.10	34.00	x	27.10	52.30	16.87
h3 second ping	0.64	0.87	x	0.36	0.73	0.64
h4 first ping	39.10	18.45	10.45	x	42.40	8.56
h4 second ping	0.69	0.88	0.35	x	0.83	0.75
h5 first ping	11.10	13.01	22.07	7.01	x	14.53
h5 second ping	1.03	0.97	0.74	0.64	x	0.36
h6 first ping	27.10	17.50	29.20	27.61	32.10	x
h6 second ping	0.90	1.20	0.64	0.66	0.36	x

Table I presents and analyses different ping time values during the communication between hosts. The ping time is the reaction time during connections between all hosts. The values represent how much time it takes to get a response after a request has been sent. Ping is measured in milliseconds (ms). In our case, the first packet always takes more time to be delivered then the second. After that, the communication stays stable and ping time becomes smaller. The first packet takes so much time because the controller must take the decision how to treat this communication. Comparing to the traditional model, in SDN environment the controller needs to recalculate each path and to put it in the tree. Once it has passed through the initialization phase, it has the knowledge how to reach the destination. After the first ping, information is available in the Flow Table, such as source IP addresses and MAC addresses. Looking at the results obtained from the testing scenario, we can conclude that delays can be caused also by the distance and the number of hops to reach the destination. Ping time increases due to the number of hops needed for packets to be delivered. Figure 6 shows a graphical representation of the results from Table I. The ping times are located on the ordinate axis in milliseconds. All sets of pings are presented on abscissa axis with different colors.

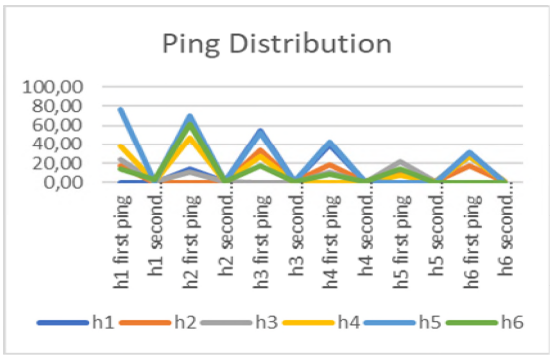

Figure 6. Ping distribution

A major conclusion for this topology, based on all tests that we have performed, is that the first ping takes more time than the following pings, and the delay depends on the number of hops to the destination. As a result, it could be concluded that this type of topology is not scalable due to the increased ping time. This reflects directly on the communication between hosts.

B. Loop Elimination security function

We are presenting in this paper the model using a loop free environment by applying Loop Elimination security function based on spanning tree. The spanning tree function is used as preconfigured virtual security function by the POX controller [10]. With the VSF, our system allows to build a loop free environment. Based on this approach, we are increasing the security by assuring availability and eliminating the possibility of loops in the network. Spanning tree will be used for preventing Storm attacks [4]. It is worth noting that data storm is excessive transmission of broadcast traffic in a network. By creating the spanning tree, we are blocking the port that can reduplicate the traffic and we are creating a safe route to each destination. For our model, we will use the topology in Figure 7. i.e., POX controller (c0), four switches (sN) with redundant links and four hosts (hN) for testing purposes. We are applying ICMP as a protocol for testing purposes, but the same results can be achieved with TCP internet-based protocol. Hosts are representing servers, as shown in the figure.

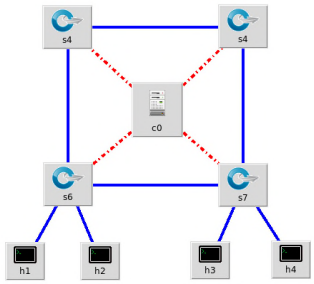

Figure 7. OpenFlow infrastructure

On this topology, we observe that the initial communication will take more time to select the best path, as it needs to eliminate the loops in the topology. Comparing the time in Section A (Table I) to the time using spanning tree, there is significant delay of the delivery. This delay comes from the need to recalculate the path by the Security Function. On the other hand, if a standard exchange protocol is used on the same topology, without a spanning tree, the virtual function will create a loop and hosts will not able to communicate with each other. In the presented console output in Figure 8, it is visible that none of the hosts are reachable.

```
mininet> pingall
*** Ping: testing ping reachability
h2 -> X X X
h3 -> X X X
h4 -> X X X
h1 -> X X X
*** Results: 100% dropped (0/12 received)
```

Figure 8. Host reachability problems

Consequently, the communication system will have a deadlock and all communications will be blocked; also, the packets will be dropped. On the controller (c0), we can register a huge amount of traffic, which overloads the

controller, as presented in Figure 9. The same packet is transmitted many times on different ports and creates a loop.

```
[openflow.of_01          ] [00-00-00-00-00-03 4] connected
[forwarding.l2_learning ] Same port for packet from 92:1a:dc:16:d6:3d
on 00-00-00-00-00-01.2.  Drop.
[forwarding.l2_learning ] Same port for packet from 92:1a:dc:16:d6:3d
on 00-00-00-00-00-04.1.  Drop.
```

Figure 9. Controller deadlock

In order to avoid loops that are blocking the traffic, we enable the Loop Elimination function. In Figure 10, we observe that by applying the security function with spanning tree, hosts can communicate with each other and avoid loops. We can see that the communications between hosts are stable and there are no dropped packets.

```
mininet> pingall
*** Ping: testing ping reachability
h2 -> h3 h4 h1
h3 -> h2 h4 h1
h4 -> h2 h3 h1
h1 -> h2 h3 h4
*** Results: 0% dropped (12/12 received)
```

Figure 10. Host reachability

The spanning tree is realized by algorithm for link detection. The controller constructs a spanning tree and then disables flooding on switch ports that are not part of the tree. This is a different process to classic spanning tree with root bridge election, presented below on Figure 11.

```
[openflow.discovery     ] link detected: 00-00-00-00-00-04.2
[openflow.spanning_tree ] Spanning tree updated
[openflow.spanning_tree ] 7 ports changed
[openflow.spanning_tree ] Spanning tree updated
```

Figure 11. Spanning tree update

Based on the Loop Elimination security function, each host is creating a table with addresses for communication between all reachable hosts. The Address Resolution Protocol (ARP) table, shown in Figure 12, presents results and provides information for MAC and IP addresses. This information can be obtained from a conventional infrastructure. Based on ARP, it is possible to discover the link layer address, such as a MAC address, which is associated with a given network layer address. This will help us to identify proper communication between hosts. For communications, the controller needs information for the hardware (MAC) address and the IP address.

```
mininet> h1 arp
Address                  HWtype  HWaddress
10.0.0.4                 ether   26:d9:4e:e9:e0:43
10.0.0.2                 ether   b2:0c:bb:90:2a:19
10.0.0.3                 ether   8a:e1:27:94:12:b6
```

Figure 12. Arp table

For clarification of our results, we present an output from the POX controller. The console output represents a Flow Table as shown in Figure 13. Using the *tcpdump* command, it is possible to observe useful information about the transferred packets. The description of the contents of packets on a network interface is defined with the *tcpdump* command.

```
mininet> dpctl dump-flows
*** s3 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=311258.123s, table=0, n_packets=0, n_bytes=0, idle_age=655
34, hard_age=65534, priority=32769,arp,dl_dst=02:00:00:00:be:ef actions=CONTROLL
ER:65535
 cookie=0x0, duration=311258.149s, table=0, n_packets=101571, n_bytes=4164411, i
dle_age=0, hard_age=65534, priority=65000,dl_dst=01:23:20:00:00:01,dl_type=0x88c
c actions=CONTROLLER:65535
```

Figure 13. OpenFlow table

During our experiments, an OpenFlow table in Figure 13 has been built and we observed that the entire communication passes over a virtualized environment. There is a link between an OpenFlow table and a Loop security function for selecting the best path. In case of a faulty link or a loop, a new path will be selected by the Loop elimination function. Looking in to the stream from the controller, by using OpenFlow as a protocol, we can analyze different data from the Flow Table, as shown in Figure 13. Based on this output analyses, we can obtain the set of MAC addresses, IP addresses, protocols and Ports. Currently, we are testing with ICMP packet, but we could also utilize TCP or UDP. By detecting a loop, we can also add additional levels of security by assuring availability for all hosts in the segment. However, observing not only IP addresses, but MAC addresses as well, allows for more flexibility. Looking at the example with the STP [2], a specific role for each port can be assigned for avoiding man-in-the middle attack, i.e., where the attacker will try to present himself as a root bridge. In this case, the port that can create a loop is blocked and there are no possibilities for deadlock. This attack targets the actual data that flows between endpoints, compromising the confidentiality and the integrity of the data. For instance, it gets banking credentials or other sensitive information. Based on the information that we can get from the flow table, we can create different security functions and apply them directly on the controller. OpenFlow provides flexibility using only virtualized environment to orchestrate the entire traffic without need of adding other devices. IDS, IPS, stateful firewall, Web proxy can be replaced by virtual security functions that are deployed at the Application layer of one controller, as presented in the framework diagram (see Figure 2).

VII. CONCLUSION AND FUTURE WORK

Using this framework, we have proved that SDN can be used on virtual environments. This will give us the possibility to apply the solution in complex environments. By creating spanning tree functions, we have assured availability on our network segment and storm attack has been eliminated. During our experimental results, we tried implementation with different topologies. During those tests, we had many failures on connections creating dead locks and scrutinizing the system. By applying this framework, the following security problems can be addressed: DDoS protection; geolocation protection and ping of death protection. They are examples that can be used as Virtual Security Function(s). All these functions can be created as future work, based on the proposed framework. Each security function will have its own assignment; therefore, many functions can be run in parallel in order to assure better

 ISBN: 978-1-61208-729-0

throughput. The utilization of VSF gives the possibility to every programmer to assign the needed level of security. This will allow the user to avoid using different high-level vendor solutions. All those functionalities will be taken by the controller (c0 in Figure 7). In addition, different types of controllers can be used for performance analyses, such as Floodlight or OpenDaylight. Web server and other type of protocol can be added in the research. As future work, the proposed framework based on SPT security function and SDN will allow us to solve very challenging problems on current the network, such as botnet detection and several types of DDoS identification. It allows to find a solution for avoiding complex attacks such as Mirai botnet and Github attack.

As a conclusion, in the paper, we identified that one of the current problems in SDN is availability. Based on the proposed framework, using POX controller, we are controlling the dataflow by creating different security functions. The Loop Elimination security function is using a spanning tree as protocol and is creating single security function for availability. This framework can not only work in real-time, it is also capable of eliminating loops. This is achievable based on the controller possibility to orchestrate the entire network segment. Performance analysis has been performed on different topologies. The main goal of this framework was to eliminate loops as a possible way of protecting from network attacks. In the future, different security functions can be created, such as DDoS elimination, IPS and many others.

ACKNOWLEDGMENT

The authors acknowledge support from the project UNITe BG05M2OP001-1.001-0004/28.02.2018 (2018-2023).

REFERENCES

[1] N. El Moussaid, A. Toumanari and M. El Azhari, “Security analysis as software-defined security for SDN environment”, 2017 Fourth International Conference on Software Defined Systems (SDS), Valencia, Spain, 8-11 May 2017

[2] Z. Yang and K. Yeung – “ILP formulation for controller tree design in SDN”, 2017 IEEE 18th International Conference on High Performance Switching and Routing

[3] Open Virtual Network (OVN) – Sflow - Telemetry, analytics, and control with sFlow standard - September 21, 2015

[4] V. Reddy and D. Sreenivasulu – “Software Defined Networking with DDoS Attacks in Cloud Computing”, International Journal of Innovative Technologies.

[5] R.S. Braga, E. Mota end A. Passito „Lightweight DDoS flooding attack detection using NOX/OpenFlow“, Proceedings of the 35th Annual IEEE Conference on Local Computer Networks, in: LCN, 2010.

[6] M. Canini, D. Venzano, P. Peresíni, D. Kostic and J. Rexford, “ANICE way to test OpenFlow applications”, USENIX Symposium on Networked Systems Design and Implementation, 2012

[7] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P.Sharma, S. Banerjee and N. McKeown “Elastictree: Saving energy in data centernet works”, Proceedings of the 7th USENIX Symposium on Networked Systems Designand Implementation, NSDI, 2010.

[8] A. Nayak, A. Reimers, N. Feamster and R. Clark, “Resonance: dynamic access control for enterprise networks”, Proceeding sof WREN, 2009.

[9] T. Muciaccia and V. Passaro, “Performance characterization of a novel DWDM all-optical SDN-like metro-access network” , 2016 24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)

[10] SDN Architecture Overview - Stanford OpenFlow Deployment https://openflow.stanford.edu/display/ONL.html (accessed July, 2019)

[11] R. Miller, “The OSI Model: An Overview”, SANS Institute InfoSec Reading Room.

[12] R. Sherwood, G. Gibb, K.K. Yap and G. Appenzeller, “Can the production network be the test bed”, Proceeding sof USENIX Operating System Designand Implementation, OSDI, 2010.

[13] S. Shin and G. Gu, “Network security monitoring using OpenFlow in dynamic cloud networks”, 7th Work-shopon Secure Network Protocols (NPSec’12), collocated with IEEE ICNP’12, 2012.

[14] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu and M. Tyson, “Modular composable security services for software defined networks”, in: 20th Annual Network and Distributed System Security Symposium (NDSS’13), 2013.

[15] C. Yoona, T. Parka, S. Leea, H. Kanga, S. Shina and Z. Zhang – “Enabling security functions with SDN”, A feasibility study - ELSEVIER

[16] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J.Rexford, S.Shenker and J.Turner, “Enabling innovation in campus networks”, SIGCOMM Comput. Commun. Rev. 38(2008)69–74.

[17] A. da Silva, C. Machado, R. Bisol, L. Granville, A. Schaeffer-Filho “Identification and Selection of Flow Features for Accurate Traffic Classification”, 2015 IEEE 14th International Symposium on Network Computing and Applications

[18] D. Hyun, J. Kim, D. Hong and J. Jeong, “SDN-based network security functions for effective DDoS attack mitigation” - Information and Communication Technology Convergence (ICTC) 2017

[19] N. Bawany, J. A. Shamsi, “DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions” - Computer engineering and computer science

[20] ISO 27001- https://www.iso.org/standard/73906.html (accessed July, 2019)

[21] J. Deng et al., “An NFV/SDN Combination Framework for Provisioning and Managing Virtual Firewalls.”, IEEE Conference on NFV and SDN, 2015.

[22] C. Munecas and A. Tirados, “Intrusion Detection Systems In Sdn-Based Self-Healing Pmu Networks”

[23] Snort IDS https://github.com/pratiklotia/SDN-Intrusion-Prevention-System-Honeypot

[24] S. Mustafiz, F. Palma, M. Toeroe and F. Khendek, “A Network Service Design and Deployment Process for NFV Systems” - 2016 IEEE 15th Symposium on Network Computing and Applications

[25] Virtual box installation https://www.virtualbox.org/wiki/Downloads (accessed July, 2019)

[26] S. Kaur, J. Singh and N. S. Ghumman, “Network Programmability Using POX Controller”, Computer Networks 147, October 2018

[27] K. Kaur, J. Singh and N. Ghumman, “Mininet as Software Defined Networking Testing Platform” – 2014

[28] PuTTY, [Online] www.putty.org (accessed July, 2019)

[29] Xming, [Online] https://sourceforge.net/projects/xming/ (accessed July, 2019)

[30] K. Atwal, A. Guleria and M. Bassiouni, “A Scalable Peer-to-Peer Control Plane Architecture for Software Defined Networks” - 2016 IEEE 15th Symposium on Network Computing and Applications

[31] S. Seeber, L. Stiemert, G. Rodosek, “Towards an SDN-enabled IDS environment” 2015 IEEE Conference on Communications and Network Security (CNS)

[32] A. Rehman, F. Siddiqui, J. Khan and M. Saeed, “Spanning Tree Protocol for Preventing Loops and Saving Energy in Software Defined Networks Along with Its Vulnerability and Threat Analyses.”, Advances in Intelligent Systems and Computing, vol 857.

 ISBN: 978-1-61208-729-0

The Benefits of a Functional Approach to Detecting and Mitigating a DDoS Attack

Robert McAndrew, Stephen Hayne, and Haonan Wang
Colorado State University
Fort Collins, Colorado, USA
email: robert.mcandrew@colostate.edu, stephen.hayne@colostate.edu, wanghn@stat.colostate.edu

***Abstract*—Distributed Denial of Service (DDoS) attacks have received significant global attention because they are increasing in frequency and severity. We analyze all flows surrounding the Network Time Protocol (NTP) amplification attack that occurred during January of 2014 at a large mountain-range university. We present an unsupervised machine learning data-driven approach that can detect and mitigate attacks in near real-time. Our method is based on thresholding, Functional Principal Component Analysis, and K-means clustering (with tuning parameters for flexibility), which dissects the dataset to reveal several categories of outliers. Using eigenfunction scores, clustering, and individual IP behavior summary statistics, we assign risk probabilities to the outliers, which enables creating dynamic firewall rules. We demonstrate the speed and capabilities of our technique in a forensic replay of the NTP attack. We show that we can detect and attenuate the DDoS within two minutes with significantly reduced volume throughout the six waves of the attack.**

Keywords-anomaly detection; clustering; DDoS; Functional Principal Component Analysis; network monitoring.

I. INTRODUCTION

There is an abundance of network security events, and one of the most impactful is the Distributed Denial of Service (DDoS) attack, in which attackers attempt to flood the target systems with huge amounts of traffic from many compromised systems leading to an interruption of the victim's services. Often, victims are high profile networks in companies, banks, or governments, and sometimes entire Internet Service Providers (ISPs) are targeted. Adversaries want to not only steal data (for later use or sale), but also disrupt operations of those targeted and impact their reputation. Hackers also increasingly use DDoS attacks as a smokescreen or distraction for more covert operations that allow them to carry out data breaches [1].

DDoS have been reported in the 1Tb/s range, driven by more than 150,000 compromised Internet of Things (IoT) devices, such as DVRs and security cameras [2]. Just a few months before that attack, the same botnet launched another in the range of 600Mbps - the volume trend is upward. Also, in Q4 of 2017, 67% of DDoS targets were blasted with more than one attack - an increase of 10% from Q3 [3]. In Q4, 32% of targets had between two and five assaults aimed at them, 6.5% that attracted between six and nine attack attempts, and a truly unfortunate 29% that were targeted over ten times (mean is 8.7 attack attempts per target over the course of the quarter). Perhaps a silver lining is that the attack duration has significantly decreased from an average of five days in 2016, to 1.3 hours at the end of 2017. However, direct costs to large organizations range from $50,000 to $100,000 per hour [4].

One variant of DDoS is the amplified reflection attack. There are several services that are vulnerable, and the one we will focus on here uses the Network Time Protocol (NTP). In this type of attack, adversaries send relatively small queries spoofing victim's Internet protocol (IP) address(es) to public servers (e.g., an NTP server), requesting a response - usually a large amount of data. As a result, this floods both the server's and the victim's network bandwidth. In 2014, 85% of all DDoS attacks larger than 100Gbps were using NTP amplification [5], and the bandwidth consumed peaked at 1% of all global Internet traffic (in late 2013 and early 2014) [6]. NTP had grown from .001% in early 2013; a dramatic three order of magnitude rise in both absolute and relative terms. This translated into organizational and financial impact [7] [8]; in fact, our own university suffered significantly. After peaking globally on February 11th (2014), NTP traffic declined back to .1% by May, still two orders of magnitude higher than at the start, as attacks continued on unmitigated systems.

In a Department of Homeland Security (DHS) funded project called "*NetBrane*" (see Figure 1), we model and characterize traffic both prior to and during DDoS attacks in order to quickly detect them and mitigate their impact. While hosted cloud-based security services offer some protection from DDoS, current solutions cannot benefit everyone. Many institutions, such as government, military, and financial organizations, need to tightly control their data, which is incompatible with cloud services. To bridge this gap, [9] is designed to be a defense service that takes advantage of the desirable properties of cloud technologies but allows customers to keep their data local. In this system, anomaly detection analytics using machine learning occurs on pre-attack network flows (inside the red box in Figure 1).

At our large university, we have installed optical taps to capture network flows at line rate (40gbps or more, top left of Figure 1) and push those flows into Hadoop Distributed File System (HDFS). Our analytics engine reads those flows in one-minute intervals, and searches for anomalies that should be investigated further. We use multi-core (parallel R packages) techniques.

In this paper, we study NTP traffic flows captured at our organization during the real 2014 main reflection attack. We conduct forensic re-analysis using our methodology to detect outliers in the flow data and apply the result to mitigate the effects of the actual DDoS in near real-time. Specifically, we detect unusual behaviors in two steps: (1) Functional Principal

 ISBN: 978-1-61208-729-0

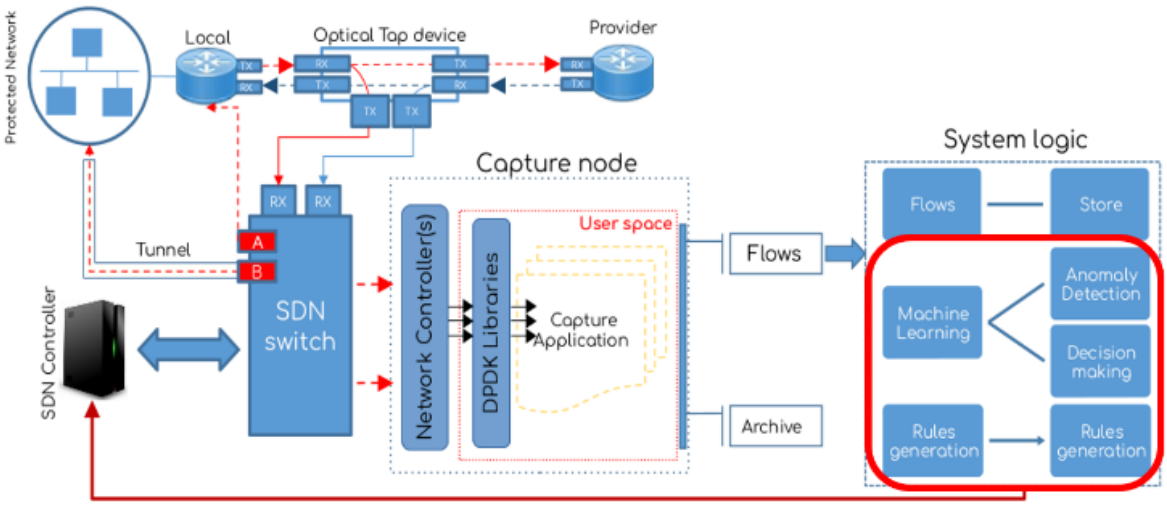

Figure 1. System Architecture

Component Analysis (FPCA) combined with (2) k-means clustering.

The remainder of this paper is structured as follows: Section 2 describes previous work related to Internet anomaly detection and Section 3 describes the dataset we use. In Section 4, the methodology of our technique is detailed, and Section 5 shows and discusses results of the analysis. Lastly, Section 6 discusses results and limitations, and Section 7 provides a conclusion.

II. Related Work

Anomaly detection methods can be classified into (1) signature based and (2) profile-based [10]. Signature-based methods use prior knowledge about characteristics of the anomaly of interest to identify suspects, and have several concerns, such as the need for labeled data, an external supervisor, and prior results from anomalies. Many machine learning classification techniques are supervised, meaning that they need to be trained on a set of labeled data prior to use. Examples of popular approaches are the Support Vector Machine, Bayesian Networks, Neural Networks, and Discriminant Analysis (surveyed in [9][10]). While these have been shown to perform well in certain situations, the reliance on labeled data can be a difficult hurdle to overcome. For the case of network traffic classification, ground truth knowledge may not be available. These supervised techniques can then only be applied when the true labels are approximated. Training on an incorrectly labeled dataset can greatly skew results [11].

In the case of a real-world DDoS attack, knowledge of which behaviors are malicious is not known; we do not have labels. Thus, supervised techniques cannot be applied. Profile-based methods create representative normal traffic behavior, and anomalies are detected by deviations from this profile. While there may be higher false alarm rates, profile-based methods are more promising due to their data-driven flexibility and they may also detect previously unknown anomalies [11]. Principal Component Analysis (PCA) is a widely used profile-based method which has been applied to detect traffic anomalies in DDoS data by decomposing network traffic into two components [12]. The anomalous subspace, which is noisier and contains the significant traffic spikes, is separated from the normal, which is dominated by predictable traffic. An individual observation is deemed an anomaly if its projection to the anomalous subspace is large. A two-stage approach was proposed, using (1) PCA to identify potential anomalies, and (2) a meta-heuristic to group them [13].

However, the use of PCA has been criticized due to issues pertaining to (i) false positive rates, (ii) traffic measurement aggregation, (iii) normal subspace pollution, and (iv) correct anomaly identification [14]. The third is important, as it highlights the need to choose which principal components represent normal behavior, and which ones represent the abnormal. It has been demonstrated that some traffic captures do not lend themselves to this partition/selection; that is, all principal components contain abnormal behaviors, and thus this approach is not usable.

Clustering is another example of a profile-based method. Clustering has been applied to all traffic, comparing the centers of known normal traffic clusters to the centers of actual traffic, to try and determine if the actual traffic is not normal [15]. Unfortunately, this approach has only been applied to Simple Network Management Protocol (SNMP) objects, not network flows, and requires known normal traffic data. Clustering techniques have been used to characterize DDoS attack traffic (k-means, CLARA, and Self Organizing Maps) [16]. K-means was found to be the most accurate for attack detection because attack traffic displays strong similarity as opposed to the heterogeneity of normal traffic. Note their attack cluster still mixed legitimate traffic in with malicious (between .4% and 2.04%). We believe this phenomenon can be eliminated by clustering only demonstrated outliers, not all traffic.

To avoid the concerns with PCA and clustering when applied separately, we will use FPCA (instead of PCA) and

 ISBN: 978-1-61208-729-0

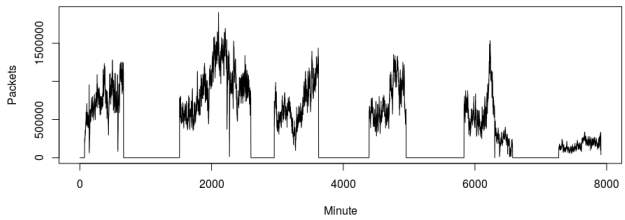

Figure 2. NTP DDoS Attack - All Waves

apply clustering to the resulting outliers [17] (that paper examined scanner behavior, where here we analyze a DDoS attack). We perform classification only using the data that is given as input, making this technique well-suited for dealing with an attack in real-time. We suggest this is more appropriate than using a supervised approach trained on data from a previous attack, as there are a wide variety of different perpetrators with attack methods, and what was previously learned may not apply.

III. DATASET DESCRIPTION

The raw data we consider in this paper is a collection of bi-directional flow records to and from our mountain-west university, relating to the NTP service. We focus on traffic between January 12 and January 25 of 2014, during the second half of which a true real-world reflection DDoS was carried out (starting in the early morning of January 18). This attack impacted the university in six waves (see Figure 2 for a plot of packet counts), with a wave defined by significantly decreased packet volume, or the monitoring system becoming unavailable. Discussion of our analytics will begin with the first wave because detection and mitigation are most crucial at the start of the attack but will generalize to all waves.

The flow records contain a plethora of useful information, such as timestamp, source and destination IP (SIP & DIP), source and destination port, packets, and bytes. We group information into one-minute bins, and the full dataset covers roughly twenty-thousand minutes. As this is a real-world dataset, we do not have ground truth knowledge of which SIPs are the victims (spoofed by attackers). However, we suggest that ground truth is not necessary as we know that a reflection DDoS indeed occurred, and we are only seeking ways to alleviate damage.

IV. METHODOLOGY

We conduct our analytics in near real-time by replaying the actual NTP attack with a sliding-window mechanism; to illustrate: the initial round of analytics is carried out on the data that appears only in the first thirty minutes of the dataset, the next round of analyses on the second to thirty-first minute of the dataset, and so on.

In each thirty-minute window, we construct a time series of contacts made for each SIP that appeared on the network, where a contact is defined to be a SIP sending at least one packet to a DIP. So, every external IP has a corresponding series with each value counting the number of DIPs they contacted in that minute. These series are used as input for Functional Principal Component Analysis (FPCA), which we use to identify outlier IPs - these are SIPs that interact with our network in an unusual way. These outliers are then clustered with the K-means algorithm to facilitate understanding of the outliers and streamline creation of firewall rules if an attack is detected. When not under attack, these outliers are displayed for operators to monitor or investigate further. Brief details of FPCA and K-means are included in Section IV.B, with full discussion found in [17].

We also monitor the time series of aggregated (SIP and DIP) packets and bytes in each thirty-minute window to volumetrically detect when the attack begins. In each window, a threshold is calculated, and when a new minute's data exceeds the pervious window's threshold, it identifies a potential start of a DDoS attack (next section).

If our volumetric threshold(s) are exceeded, we suspect that we may be under attack. At this point, the sliding-window becomes a growing window, fixed at the current time, and subsequent minutes are appended to the previous data. For example, if a significant volumetric change is detected while we are considering a window from minute 30 to 59, the next window we analyze will contain data from minute 30 to 60. This growing-window is used so we do not skew outlier detection as the attack proceeds; that is, if more behavior enters the network that is similar to that which caused the volumetric trigger, we do not want it to become representative of normal traffic.

When we are under attack, the outliers gathered and clustered by FPCA+K-means are remembered in what we refer to as a total recall strategy. We keep a running list of outliers that are detected and assign a threat level to each based on the individual IP's activity on the network (details in Section IV.B). From the threat levels, we construct a list of suspected attackers, as well as a list of those that are believed to have acceptable behaviors. It may seem counter-intuitive that acceptable traffic can be flagged as unusual, and more discussion on this is given in Section V. These two groups can then be blocked from and allowed into the network, respectively, to mitigate the attack and yet allow some known good systems to continue access.

As a final note, when the analytics have detected an attack, we implement a two-pass procedure where we repeat FPCA with outlier collection on the subset of the data that were classified as non-outliers from the first pass. The outliers from the first and second passes are added to the running list. This two-pass is carried out only during attack mitigation, and later we will compare the effects of one and two passes to demonstrate the marginal gain from each round of outlier collection. (Discussed in Section V).

All data management and analytics are carried out in version 3.4.4 of the R programming language. The analytics in each sliding-window iteration takes approximately four seconds, while the attack analytics (growing-window) take no more than ten seconds (on a 10Gb set of flows) using eight cores.

A. Volumetric Attack Detection

As a first warning for a DDoS event, we seek to identify a drastic increase in packets or bytes sent and received by the network in any given minute. We define this drastic increase to be an instance when a new minute's data exceeds a

 ISBN: 978-1-61208-729-0

threshold from the previous thirty-minute window. Specifically, we calculate separate thresholds for packets and bytes in each window and compare the new minute's aggregated packet and byte counts to these thresholds. The threshold is defined by (1),

$$Threshold = \max_{t \in H} X_t + cv \cdot SE\left[\max_{t \in H} X_t\right]. \quad (1)$$

In (1), X_t for $t \in H$ is the time series of packets or bytes in the given window of history. $SE\left[\max_{t \in H} X_t\right]$ is the standard error of the maximum packet or byte count from a LOESS fit of the packet/byte time series in the window of history. Lastly, cv is a critical value determined from investigation of long-term (months) packet and byte distributions. This takes the largest value in the given window of history and sets a threshold greater than it by adding a scaled measure of this maximum's variability. That is, we start with the largest packet or byte count in local history which is considered acceptable because it previously did not indicate a potential DDoS, then raise this to calculate the threshold. This is motivated by the idea that we may see normal network activity that is larger than the previously accepted amount, but we only expect a DDoS if new packet or byte counts exceed what we expect from historical variability of our data.

If a new minute of data is collected and it does not exceed the previous window's threshold, the time series of flows for the new thirty minutes are stored, and the threshold is recalculated. This creates a dynamic threshold for volumetric detection that takes usual network activity into account.

If a new minute of data is collected and either our packet or byte threshold is exceeded, we consider an attack to be starting and we initiate the growing window and two-pass FPCA outlier collection.

B. Outlier Detection and Risk Assessment

FPCA takes as input a collection of series that can be treated as realizations of a function over time, and then models these series as a mean curve plus a linear combination of eigenfunctions. These eigenfunctions are orthogonal curves created by finding the largest dimensions of variability in the data. That is, the first eigenfunction can be thought of as the direction of highest variance, the second captures the next most variance, and so on. When the original series are projected onto the eigenfunctions it produces scores, which are the locations of the observations on each new dimension. These scores are used to identify data points as outliers. For each eigenfunction, we calculate the bounds $\bar{x} \pm ks$, where $\bar{x}$ is the average score, s is the standard deviation of the scores, and $k > 0$ is a constant. If a score is outside of the bounds on any eigenfunction, it is flagged as an outlier. As we use the time series of contact counts for input, these outliers are the IP addresses that are interacting with the network in an unusual way. With the feature of descending variance in the eigenfunctions, SIPs that are outliers based on our definition are also extreme in the sense of the original dataset. For our analytics, we use the Principal Analysis by Conditional Expectation (PACE) implementation of FPCA [18].

The number of eigenfunctions to use in the FPCA model is a parameter that must be selected so we use the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) [19]. Should the results of these not match, factors specific to the situation should be considered to determine which is better suited [20]. As this analysis is carried out in each iteration of the sliding window, disagreement between AIC and BIC at some point is likely. For the purposes of detecting unusual behavior in network traffic, the true model for which can be highly complex, we focus on AIC. It can also be the case that AIC and BIC are not applicable, depending on the amount of data available and its sparseness. In these situations, the number of eigenfunctions is chosen using a Fraction-of-Variance-Explained (FVE) cutoff. That is, we select the first n eigenfunctions that capture a certain percent of the variance from the original dataset. This method is always applicable, provided the FPCA model can be used successfully.

Following the gathering of outlier SIPs on the network in the available window, they are clustered using the K-means algorithm of [21]. The number of clusters is also chosen using a FVE cutoff commonly referred to as the elbow method. The cluster amount chosen is such that adding one additional cluster will not significantly increase how much variance of the original dataset is explained by our clustering; i.e., the point of diminishing marginal returns. The outlier SIPs are clustered based on their proportion of successful contacts, where a success is defined to be at least one packet sent back to the SIP by the DIP being contacted. With this completed, the result is a set of SIPs that are interacting with the network in an unusual way, stratified by their success. This facilitates easier understanding of the traffic that is detected.

After the clustering, we assign a threat level (TL) to the outlier IPs. This is calculated using (2).

$$TL = \alpha_1 v + \alpha_2 d + \alpha_3 (1 - c). \quad (2)$$

In (2), the α_j's are constants that satisfy $0 < \alpha_j < 1$ and $\sum \alpha_j = 1$. Further, v is the proportion of volume sent and received by the given IP relative to that of the entire window, d is the number of destinations contacted by the IP divided by the total number contacted by the SIPs in the given sliding window iteration, and c is the fraction of minutes that the SIP reached out to at least one destination. For purposes of the analytics in this paper, we use $\alpha_j = 1/3$, but other choices can be made based on context-specific factors. With these definitions, the threat level TL exists between 0 and 1 and represents the maliciousness of the outlier - a threat level closer to 1 indicates greater likelihood the IP is malicious. The quantities used in this calculation are chosen based on analysis of NTP behaviors prior to and during the attack. It is likely that different factors must be considered when assigning a threat level to behaviors on other services.

Since we only group the outlier SIPs based on one numerical summary of their behavior, there can be instances

 ISBN: 978-1-61208-729-0

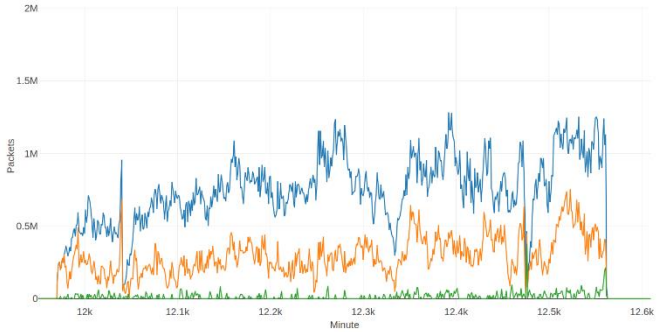

Figure 3. NTP reflection DDoS wave 1: actual packets (blue), one-pass reduced packets (orange), two-pass reduced packets (green)

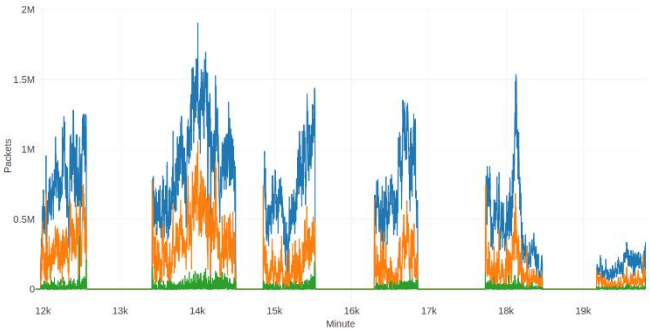

Figure 4. NTP reflection DDoS: actual packets (blue), one-pass reduced packets (orange), two-pass reduced packets (green)

where behaviors are mixed within clusters. The TL's alleviate this issue, as we can search for non-threatening SIPs in lower-centered clusters or threatening ones in the high.

V. RESULTS

We first present results from attack detection, and then turn to mitigation. We begin our sliding-window analysis by working with the section of the dataset prior to the attack. This is roughly a week's worth of data, during which our detection mechanism does not trigger. So, in each iteration of the window, we carry out only one pass of outlier detection (FPCA) and the outliers are clustered with K-means. This results in 95 distinct outliers out of approximately 6 thousand SIPs across the entire pre-attack period. When assigning threat levels (TL's), the vast majority of these non-attack TL's are close to 0, indicating that the behavior is non-threatening. In fact, the outliers we identify here have well-known and acceptable NTP behaviors - they are consistently checking the time with the network's published NTP servers. We think of such behaviors as active peers because they are working in a way we expect for this service [22]. These are identified as outliers because they are contacting the network in a way that is unusual with respect to the rest of the dataset considered; that is, reaching out to one DIP uniformly over time.

In each iteration of our thirty-minute window prior to the attack, our threshold is calculated as in (1), with cv calculated from long-term historical distributions of maximum packet and byte counts. Specifically, we recreated our sliding window procedure on approximately 5 weeks of data directly prior to the dataset described in Section III, saving the maximum packet and byte counts in each window. We then find the 99th percentile for both distributions and standardize them to calculate cv. For example, we find the 99th percentile of our maximum packet counts, then subtract off the mean maximum packet count and divide this by the standard deviation of the distribution. The same is done for byte counts. In the case of packets, cv is 19.94, and in the case of bytes, it is 21.2.

As the sliding-window marches forward, we eventually reach the minute at which the attack begins (12000 minutes into the dataset). In the iteration that captures the start of the event, it is as if the attack has been going on for one minute, and we are observing that first minute along with the previous thirty (of pre-attack traffic). With a new minute of data, we compare the packet and byte counts to previous thresholds and find that both are exceeded. Our attack flag is triggered, and we begin to grow our window of history, which now contains the 30 minutes prior to the attack, as well as its first minute. As an approximation of real-world monitoring, our method detects the DDoS attack during its second minute of activity.

Once the attack has been detected, we begin applying our two-pass procedure, using the 3-sd from the mean cutoff to define outliers in each pass ($\bar{x} \pm 3s$ on each eigenfunction). With outliers identified, TL's are assigned and stored before adding the new minute's worth of data to our now growing-window (because we are under attack). In this new minute, we simulate blocking the high-threat outlier IPs activity from the network; that is, the data generated by all malicious outlier SIPs gathered in previous window iterations are removed from the new minute's information. This mimics the creation of firewall rules that would block IP addresses from the network and is the proposed mitigation strategy for DDoS - we remove those outliers which are found to be the malicious actors in the attack. Figure 3 visualizes the packets counts and mitigation during each pass, in order to demonstrate the reduction gain from each round of outlier collection. At the end of the first wave, we have identified about 100 unique SIPs as outliers. The one-pass reduced volume is approximately 60% less than the actual first wave, while the two-pass reduced volume attains a reduction of 95%. This is a significant mitigation result, making the attack look much more like pre-attack traffic than a DDoS.

Later waves of the attack are handled in the same manner, and Figure 4 visualizes the reduced packet volumes. Approximately 3000 SIPs are identified as outliers throughout all six waves. The volume reduction achieved in the first wave extends to the entire attack: 95% of the overall volume is masked by removing the traffic from outliers (detected by two passes of FPCA) that are determined to be threatening.

While we block the traffic from the threatening outliers, we propose allowing activity from the active peers through to the network. The series of packet volumes with and without the non-threatening activity are virtually identical, with the active peer outlier traffic representing only 3% of the overall data. This gives high mitigation levels of the attack while allowing the known active NTP actors (from clustering) to continue operating. We acknowledge that malicious actors could possibly take advantage of this but will address the issue

 ISBN: 978-1-61208-729-0

in future work. For any of the window iterations during the attack, no more than six clusters are used at any point (as set by the elbow method). This stratification consistently collects what we consider as active peer behavior in the higher-centered clusters (those with the largest portion of success), facilitating the creation of firewall rules.

We now turn to alternatives in parameter choices and investigate their effects on mitigation. The standard deviation (sd) threshold can be altered. $k = 3$ was previously used, and we also considered $k = 1$ and $k = 2$. On one-pass of outlier detection, the 2-sd cutoff removes 80% of the traffic while the 1-sd achieves 90% reduction. A second-pass using these cutoffs achieves close to the 95% reduced volume found using the 3-sd threshold. This indicates that we do not gain new outliers in this attack by varying standard deviations.

Another option for mitigation would be to mask the subnets that contain the outliers found by our method. This option would reduce the number of firewall rules that need to be created to block the outliers; instead of creating a rule for each individual IP, a rule for the /24, /16, or /8 subnets could be created - we tested this. For example, if the address 169.229.70.49 is found as an outlier and we are masking traffic at the /16 level, we omit IPs with addresses 169.229.X.X from further iterations of our sliding window. Figure 5 shows the mitigation results on the first wave using one-pass (3sd) with varying levels of subnet masking. Observe that blocking the individual outlier IPs, the /24, and /16 subnets achieve similar reductions in traffic, while the /8 subnet mask diminishes packets by almost 95%. Note this uses one-pass of outlier detection, indicating that this blocking results in mitigation like our complete two-pass procedure without the subnet rules, however we suggest that blocking the subnets can lead to blocking legitimate traffic too.

Avoiding blocking of legitimate traffic is the motivation for the two-pass procedure. We know that a DDoS is happening but have no ground truth about the attackers – thus, we must at least consider the notion of false-positives. Blocking /8 will certainly block more legitimate traffic than blocking outliers from a second pass of FPCA. At each application of FPCA when scores are calculated, we can estimate the shape of the distribution of scores on each eigenfunction using kernel density estimation [23]. We carry this out on the scores from the second pass of FPCA, when the outliers detected in the first pass are removed. Specifically, we test the estimated distributions for normality using the Shapiro-Wilk test [24]. If we find evidence for the scores being approximately normal, it indicates that the IPs detected in the first pass are appropriate outliers. We can extend this idea to an n-Pass procedure, in which we stop the repeat applications of FPCA when normality of the scores is reached, or we can no longer apply our method. The most common reason is that eventually no outliers are found by FPCA.

We recall that we employ a Total Recall strategy, in which our threatening outliers are remembered from previous window iterations and masked from future incoming data. As this list grows over time, there are less outliers to be found and less passes of FPCA are needed. The second reason the procedure stops is because of no data, in which this is the case

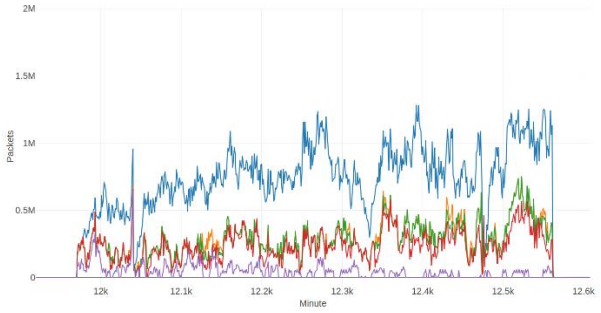

Figure 5. NTP reflection DDoS wave 1: actual packets (blue), one-pass reduced packets (orange), one-pass /24 reduced packets (green), one-pass /16 reduced packets (red), one-pass /8 reduced packets (purple)

or the data available is few and sparse. Lastly, normality is reached only a small portion of the time. The average number of passes throughout all waves of the attack is 1.88, with only 1 window iteration needing 11 passes. This occurs near the beginning of the first wave, before a large list of outliers is built up. With this, and the packet reduction achieved by our methodology, we believe two passes is appropriate for outlier detection and attack mitigation.

VI. Discussion and Limitations

The notion of false-positives arises whenever anomaly detection is discussed. Without ground truth of the attackers in the dataset we analyzed, there is no way to accurately measure the false positive rate of the results in Section V. Our assignment of a threat level attempts to alleviate this issue, as we allow the non-threatening outliers to remain in the network's traffic. Even with this, there may be a few SIPs blocked that are not malicious. We suggest that this is not a major concern when truly under attack, as the security of the network is paramount and only outliers are being blocked.

The formula for calculating threat levels is created based on observing the data during and prior to the attack. Data from different services may require a different calculation of the TL, and this is true of the NTP-port as well - as this attack and more are studied further, the TL computation will be improved. Also, the methodology will benefit if selection of the α_j parameters is made dynamic and data-driven. For example, if network history or other contextual information can help select the weights for factors being considered, threat level assignment is expected be more accurate.

A major part of our analysis relied on the sliding-window approximation of real-time streaming data. We used a fixed thirty-minute window, because it is a near worst-case scenario, in that it is the smallest window of time-series data on which FPCA can still be applied. A larger history could be kept, and different types of behaviors may become apparent. This requires more data in RAM, especially during an attack, but we believe that the attack detection and mitigation would occur in the same way as presented in Section V.

VII. Conclusion

We have demonstrated an approach to detecting and mitigating an actual DDoS attack that occurred in early 2014. Dynamic volumetric thresholding is shown to detect the

 ISBN: 978-1-61208-729-0

attack, and the FPCA+K-means approach mitigates the attack volume significantly (by >95%). These unsupervised approaches are best suited for detection and mitigation of unknown attacks. We have proposed multiple options for reducing the packet volumes of the attack, including the alteration of tuning parameters and masking subnets. Assignment of threat levels to the outliers allows for better understanding of the SIPs identified.

ACKNOWLEDGMENT

This material is based on research sponsored by the Department of Homeland Security (DHS) Science and Technology Directorate, Homeland Security Advanced Research Projects Agency (HSARPA), Cyber Security Division (DHS S&T/HSARPA CSD) BAA HSHQDC-14-R-B0005, and the Government of United Kingdom of Great Britain and Northern Ireland via contract number D15PC00205. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Department of Homeland Security, the U.S. Government, or the Government of United Kingdom of Great Britain and Northern Ireland. The research of Haonan Wang was partially supported by NSF grants DMS-1521746 and DMS-1737795.

REFERENCES

[1] S. Mansfield-Devine, "The Growth and Evolution of DDoS," *Network Security*, pp. 13 - 20, 2015.

[2] B. Krebs, "KrebsOnSecurity Hit With Record DDoS," 2016. [Online]. Available: https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/.

[3] I. Incapsula, "Global DDoS Threat Landscape Q4 2017," 2017. [Online]. Available: https://www.incapsula.com/ddos-report/ddos-report-q4-2017.html.

[4] T. Matthews, "Incapsula Survey: What DDoS Attacks Really Cost Businesses," 2014. [Online]. Available: https://lp.incapsula.com/rs/incapsulainc/images/eBook%20-%20DDoS%20Impact%20Survey.pdf. [Accessed July 2019].

[5] M. Mimoso, "Volume of NTP Amplification Attacks Getting Louder," 2014. [Online]. Available: http://threatpost.com/volume-of-ntp-amplification-attacks-getting-louder/105763.

[6] J. Czyz, M. Kallitsis, M. Gharaibeh, C. Papadopoulos, M. Bailey and M. Karir, "Taming the 800 Pound Gorilla: The Rise and Decline of NTP DDoS Attacks," in *Internet Measurement*, Vancouver, BC, Canada, 2014.

[7] K. Arora, K. Kumar and M. Sachdeva, "Impact analysis of recent DDoS attacks," *Intntl. Journal on Computer Science and Engineering*, pp. 877-883, 2011.

[8] J. Armin, B. Thompson and P. Kijewski, "Cybercrime Economic Costs: No Measure No Solution," in *Combatting Cybercrime and Cyberterrorism*, Springer, 2016, pp. 135-155.

[9] "NetBrane, Funded Project, Department of Homeland Security Award D15PC00205," 2015-2019.

[10] M. Ahmed, A. N. Mahmood and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, pp. 19-31, 2016.

[11] M. Soysal and E. G. Schmidt, "Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison," *Performance Evaluation*, pp. 451-467, 2010.

[12] A. Lakhina, M. Crovella and C. Diot, "Diagnosing Network-Wide Traffic Anomalies," in *Computer Communication Review*, 2004.

[13] G. Fernandes, L. F. Carvalho, J. J. Rodrigues and M. L. Proenca, "Network anomaly detection using IP flows with principal component analysis and ant colony optimization," *Journal of Network and Computer Applications*, pp. 1-11, 2016.

[14] H. Ringberg, A. Soule, J. Rexford and C. Diot, "Sensitivity of PCA for traffic anomaly detection," *ACM SIGMETRICS Performance Evaluation Review*, pp. 109-120, 2007.

[15] W. Cerroni, G. Monti, G. Moro and M. Ramilli, "Network attack detection based on peer-to-peer clustering of SNMP data," in *International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, 2009.

[16] B. Hammi, M. C. Rahal and R. Khatoun, "Clustering methods comparison: Application to source based detection of botclouds," in *Security of Smart Cities, Industrial Control System and Communications, 2016 Intntl. Conf. on*, 2016.

[17] R. McAndrew, M. Gharaibeh, H. Wang, S. Hayne and C. Papadopoulos, "A Functional Approach to Scanner Detection," in *Proceedings of the Asian Internet Engineering Conference*, Bangkok, Thailand, 2017.

[18] H.-G. Muller and J.-L. Wang, 2015. [Online]. Available: www.stat.ucdavis.edu/PACE/.

[19] S. I. Vrieze, "Model selection and psychological theory: a discussion of the differences between the AIC and the BIC," *Psychological methods*, p. 228, 2012.

[20] Y. Li, N. Wang and R. J. Carroll, "Selecting the number of principal components in functional data," *Journal of the American Statistical Association*, pp. 1284-1294, 2013.

[21] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C*, pp. 100-108, 1979.

[22] T. N. Distribution, "Association Management," 2014. [Online]. Available: http://doc.ntp.org/4.1.0/assoc.htm.

[23] J. S. Simonoff, Smoothing methods in statistics, 2012: Springer Science & Business Media.

[24] J. Royston, "Algorithm AS 181: the W test for normality," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, pp. 176-180, 1982.

[25] K. Labs, 2018. [Online]. Available: https://usa.kaspersky.com/.

[26] A. Singh, N. Thakur and A. Sharma, "A review of supervised machine learning algorithms," in *Computing for Sustainable Global Development (INDIACom)*, 2016.

 ISBN: 978-1-61208-729-0

Detection of a Rogue Switch in a Local Area Network

Vijay Bhuse, Andrew Kalafut and Lisa Dohn
School of Computing and Information Systems
Grand Valley State University
Allendale, MI, USA
email: {bhusevij, kalafuta and dohnl}@gvsu.edu

***Abstract*—There are many solutions available for detecting a rogue wireless access point, but none exists for detecting a rogue switch or a router in a wired environment. This is at least in part due to the implicit assumption that a wired environment is safer. In this paper, we present three solutions to detect a rogue Ethernet switch. Each of these solutions has its advantages and may be used independently of the others. We prove that our approach is practical and feasible by simulation and analysis of our solutions.**

Keywords—rogue switch; detection; network; security; LAN.

I. INTRODUCTION

The concern of potentially compromised switches and routers being installed in a network is significant enough that the United States House of Representatives Intelligence Committee investigated network hardware suppliers over the possibility that their routers could have backdoors allowing them to be compromised [1]. This risk is not only limited to specific suppliers. A recent vulnerability was found in Cisco routers that could allow an attacker to execute malicious code on the vulnerable network equipment [2].

A rogue switch is a switch that is connected to the network without the authorization of the network owner or a network administrator. Rogue switches are a threat to the security of a network. They create a very serious vulnerability, which can be exploited by an attacker to spy on the business, government or military installations. Existing solutions for preventing rogue devices on networks include the IEEE 802.1X [3] protocol (which provides an authentication system for devices requesting to connect to the network) and port security (which helps limit the devices that can connect to a switch and transmit frames). However, prevention methods may not always be successful due to a variety of reasons including lack of availability or misconfiguration. To provide defense in depth, it is extremely important to add a second line of defense by designing solutions that detect rogue switches. New techniques are needed in order to address this problem.

A rogue switch could be introduced to the network in one of the following ways.

- A compromised employee may connect their own *rogue* switch to the network and connect rogue hosts to it and can then use these hosts to launch attacks on the network.
- "Bring your own device" (BYOD) policies are also making it easier for employees to inadvertently introduce a rogue device into the network. A non-malicious user (such as, an employee at a business who prefers to use his own devices) might connect a switch or a host to the existing network, which may not have preventative security measures implemented. This switch or hosts might be insecure and may provide an avenue for hackers to access the network resources.

A rogue switch can be added only by connecting an Ethernet cable from it to an existing port of the valid switch. Rogue switches can threaten the confidentiality of messages on the network, degrade the performance of the network, or even allow unauthorized access to the network.

Adding a rogue router is much more difficult for the attacker than adding a rogue switch, as adding the router requires changes to the configuration of the valid router and need to support one or more Internet Protocol (IP) address subnets. We focus only on the problem of detecting a rogue switch in this paper, not a rogue router.

There are many solutions currently available for detecting a rogue wireless access point, but none exists for detecting a rogue switch or a rogue router in a wired environment. This is because of the *implicit* assumption that a wired environment is safer. However, if left undetected, a rogue switch can cause a number of security issues on the network.

As networks grow in both size and complexity, the detection of these rogue devices becomes even more difficult. The evidence of compromise is hard to detect because it is buried inside the traffic and complexity of these large networks. We discuss three specific solutions in this paper to detect a rogue Ethernet switch.

The rest of the paper is organized as follows. Section 2 covers related work. Section 3 covers problem definition. Section 4 discusses and analyzes our detection solutions in detail, and we conclude in Section 5.

II. RELATED WORK

There are many existing solutions to detect a rogue wireless access point [4]. Wireless traffic analysis monitors all traffic on the network to determine if any packets are suspicious. Site survey software is also available to assist with security analysis. Software tools, such as NetSpot [5] assist with the detection of unauthorized devices, traffic interference, and rogue wireless access points to the network. These solutions are not applicable to wired networks.

There is a slight possibility that the rogue switch can be detected through an IP sweep tool (such as network mapper

 ISBN: 978-1-61208-729-0

tool Nmap [6]), if information is captured about a host connected to a rogue switch (wired). These tools investigate the entire network and alert of any abnormality, at which point the task of analyzing the results and locating the physical whereabouts of the switch falls on the network administrator. However, this is no simple task, and it grows exponentially more complex as the network size increases.

The IP address of the host that is connected to the rogue switch is not singled out using the IP sweeping tool, so an investigative work is required in order for the network administrator to begin to track down the rogue switch.

There are a number of issues that complicate the process of finding a solution for rogue switch detection on wired networks. Some of these include:

- Unmanaged Layer 2 switches are not commonly traceable, because they do not have IP addresses [7].
- Neighbor discovery protocols are not typically supported on unmanaged switches. Therefore, connectivity information of neighboring switches will not be reported [8].
- Detection of a rogue switch does not necessarily reveal its physical location, which, in turn, is a completely separate and a difficult task.

All of the above issues make the task of detecting a rogue switch extremely difficult for a network administrator.

III. PROBLEM DEFINITION

The network being analyzed and tested will be similar to one which may be used by a small-to-medium size business, where all computers (used by employees) are part of the intranet Local Area Network (LAN) and are connected using Ethernet switches.

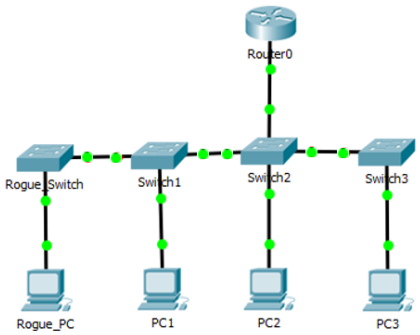

Fig. 1. Rogue switch connected to a normal network.

Figure 1 shows an example topology where a rogue switch is connected to a network. In this example, there are three switches (namely Switch1, Switch2 and Switch3), each with a single PC device connected, and a router (Router0) connected to one of the three valid switches. The rogue switch, Rogue_Switch, is connected to an open slot on one of the valid network switches (Switch 1), and there is a single rogue host (Rogue_PC) connected to the rogue switch. In this example, Rogue_Switch and Rogue_PC would not have permission to be connected to the rest of the network, though are connected anyway.

In all rogue switch cases, the rogue device is physically connected to a valid network switch without permission. This is a problem since network traffic can possibly be intercepted by hackers, who may use that information to gain access to the system and potentially use this access to steal sensitive information. Scenarios involving rogue switches can range from hackers attempting to gain access to the network, to non-malicious employees hoping to use their own devices either to gain more convenient network access or to simply use their own hardware. These devices may not be as secure and may provide an easy vulnerability for hackers to exploit. If left undetected, rogue switches or routers can cause many security issues on a network.

IV. DETECTION SOLUTIONS

We propose three unique, realistic, and independent solutions for detecting a rogue switch that is already on the network. Our solutions are based on using collected evidence for detecting *anomalies* using (a) Dynamic Host Control Protocol (DHCP) request messages, (b) Simple Network Management Protocol (SNMP) and Address Resolution Protocol (ARP) broadcast traffic detection and (c) Media Access Control (MAC) address whitelisting. These solutions act as a second line of defense that can be implemented alongside typical prevention measures. We validated our solutions by using Cisco Packet Tracker [9], a simulator that supports an extremely realistic simulation of IP networks using routers, switches and hosts. Our solutions work for routers and switches from any manufacturer or hosts with any operating system.

A. DHCP Request Message Detection

In many networks IP addresses are assigned DHCP. We can use the observed behavior of DHCP when hosts become disconnected and reconnected as a way of detecting rogue switches. This solution requires the ability to temporarily shut down a switch port connecting a switch to a host to determine if a rogue switch exists in between the two. This solution will not work on networks that assign addresses statically or otherwise avoid DHCP, but DHCP use is quite common.

Figure 2 displays an example of a "normal" network (one with no rogue devices) whereas Figure 3 shows a similar network where a rogue switch is present. We configured LAN interface Gi0/1 of Router0 and configured a DHCP server to run on Router0. PC0 receives its IP address, subnet mask and default gateway configuration from the DHCP server running on Router0.

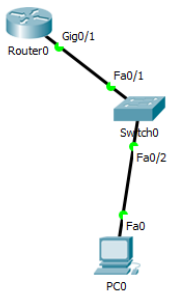

Fig. 2. Normal Network (No Rogue Switch).

In the network from Figure 2, the Fa0/2 port on Switch0 could be temporarily shut down and then brought back up, while we monitor the Fa0 interface on PC0. This process can either be automated or manual. When the switch port shuts down and comes back up, the host logs should indicate a change in the state of interface of the PC. If the interface state did not change, this indicates that the link between the host and the switch also includes one or more devices in between the two. This is because the host did not lose connection to the device it is connected to (a rogue switch, for example), so the interface retains the state.

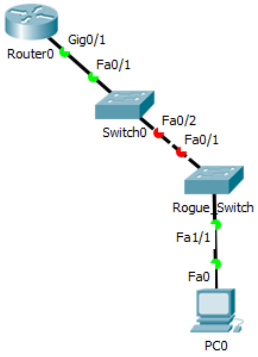

Fig. 3. Network With Rogue Switch.

Once the link is re-established, a DHCP request is triggered by PC0 if there is no rogue switch between the PC and a valid switch. There is no DHCP request if a rogue switch is present. In Figure 3, there is a rogue switch (titled Rogue_Switch) between Switch0 and PC0. When Fa0/2 port of Switch0 is shut down and brought up, the link between PC0 and Rogue_Switch retains the state. When the link between Switch0 and Rogue_Switch is disconnected, the interface Fa0 of PC0 will remain up the entire time and will not trigger DHCP request from PC0. Upon re-establishment of the link between Switch0 and Rogue_Switch, PC0 has no need to send another DHCP message. The missing DHCP message is an indicator of the presence of a rogue switch.

This detection mechanism can be circumvented if the rogue switch has ability to generate DHCP request messages and make it look like it came from PC0. In this case the switch can send the request when it becomes reconnected, as PC0 would have it the switch was not present. It can also be circumvented if the rogue switch disconnects PC0 whenever the rogue switch becomes disconnected from Switch0. However, both of these circumventions rely on non-standard behavior from the rogue switch and therefore increase the effort required from the attacker to evade detection.

In a large network, this process can be automated. The system will cycle through each switch directly connected to a host, and one by one, disconnect each link, wait, then reconnect the link, all while monitoring the interface status of hosts. If the status did not change as expected, the system will record the specific information of the link being tested, and flag the link as potentially part of a connection to a rogue device.

Link interruption can be scheduled to be completed at a convenient time (such as, at 3:00 a.m. when network usage might be minimum). This solution can also be automated, which removes much of the physical burden from information technology (IT) administrators.

DHCP is extremely time consuming, especially for large networks, since each link needs to be individually disconnected and reconnected one at a time. The larger the network, the more links that need to be tested, and the more time required to get through them all. This in turn affects the frequency of testing each link. Overall, the larger the network, the less frequently each link can be tested.

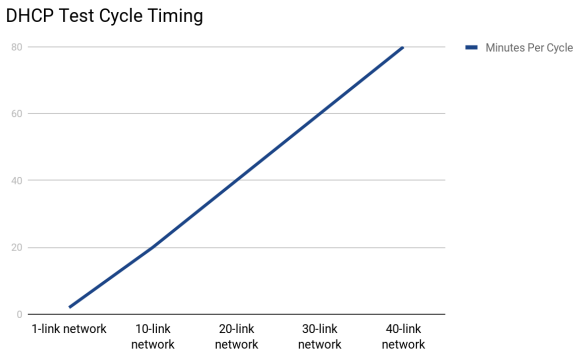

Fig. 4. Time it takes to test all switches in networks of various lengths (1-40 links).

Figure 4 displays the approximate time it takes (in minutes) to test each link in networks of various sizes. As seen in each graph, time increases linearly as the size of the network grows. This graph assumes that one testing cycle (to test all links in the network exactly once each) requires approximately two minutes to complete based on the following experimental measurements we did.

- 30 seconds time where the link is disconnected.
- 30 seconds initialization/power on time.
- 45 seconds time to monitor/verify the interface status and DHCP traffic.
- 10 seconds time to verify network connectivity re-established.
- 5 seconds time to transition to next link and tolerance.

The larger the network, the fewer options that exist for when the links can be tested. A 10-link network that takes approximately 20 minutes assuming all links are shut down and brought back one after another, so only one host is disconnected at a time. Similarly, a 1,000-link network (or larger) requires much more of a time commitment (2,000 minutes, or 33.3 hours). You can reduce the downtime significantly by shutting down multiple links at a time.

This solution will not work for hosts that have a critical need to be connected to the network at all times with no interruptions, since it requires a temporary disconnection of the link between the switch port and host. Additionally, it requires the network to be using DHCP. However, DHCP use is common, and many hosts can tolerate a very brief temporary loss of connectivity.

 ISBN: 978-1-61208-729-0

False positives (detecting a rogue switch when none is present) will be very unlikely with this mechanism. A false positive would require the DHCP request to not be received even though the rogue switch is not present. All DHCP clients must however send a DHCP request when disconnected and reconnected in case they have been reconnected to a different network. A DHCP client without this behavior would not function in very common real-world situations. Therefore, the only reasonable way the DHCP request could be missing is packet loss. However, a properly implemented DHCP client will retransmit this request several times. Therefore, a false positive will only occur if the hosts on the network have improperly implemented DHCP clients.

B. SNMP & ARP Broadcast Traffic Detection

This solution detects a potential rogue device through a multi-step process. We assume that an SNMP agent is configured on every valid switch and the switch monitors the traffic. We implemented the topology in Figure 5 within the simulator. We configured the LAN interface of Router0, PC0 and PC1. We assume that Rogue_Switch and Rogue_PC are added by an adversary.

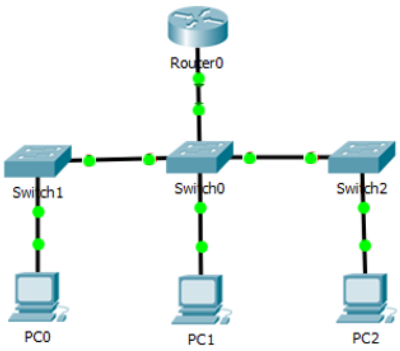

Fig. 5. Rogue_Switch broadcasts traffic. When either Switch0 or Switch1 detect the broadcasted traffic, a trap alert will be sent.

As seen in Figure 5, when an adversary adds Rogue_Switch and Rogue_PC to the network, traffic sent from Rogue_PC is broadcast by the Rogue_Switch (because the MAC Address table of Rogue_Switch would be empty). In this case, frames are broadcast to both Switch0 and Switch1. All valid switches in the network will detect the broadcast traffic and will alert the network administrator because a sudden broadcast traffic is an anomaly. There would also be ARP traffic at least at the beginning when new rogue devices are added. Once an alert is sent, the network administrator filters the ARP traffic that was generated around the time that the alert message was received. Broadcast frames and ARP traffic indicate the possible presence of a new device. The network administrator will be able to either confirm that a rogue switch or rogue device is connected or can clear the alert if the administrator is confident that the flagged traffic is permitted.

The rogue switch could be configured to prevent the detected behavior. This is only possible if the rogue switch MAC address table is already populated with correct entries. This would require an attacker to put lot of efforts and it may not even be possible to learn MAC addresses of neighboring switches or PCs. We can automate the detection of broadcast traffic using SNMP, as explained next. SNMP is designed to monitor events, and can be configured to send trap alerts whenever a specified event occurs. Almost any network monitoring software will work for this experiment, as long as broadcast traffic can be detected.

TABLE I. OIDS USED FOR DETECTION

Object ID	Name	Description
.1.3.6.1.2.1.2.2.1.12	ifInNUcastPkts	Number of inbound broadcast/multicast packets
.1.3.6.1.2.1.2.2.1.17	ifOutUcastPkts	Number of outbound broadcast/multicast packets

Table 1 shows a list of some OIDs that would be useful in this solution [10] [11]. These OIDs can be used to monitor broadcast traffic. Trap alerts should be sent when a valid network switch detects that another switch has sent broadcast traffic, indicating that the switch does not know where to forward the packet. This can help uncover a rogue switch that was recently connected.

The benefits of this solution are that the rogue switch will be detected fairly quickly after it is connected, because the forwarding table will be empty and the switch is forced to broadcast frames that it does not know where to forward. Also, before any corrective action is taken against the rogue switch, the network administrator is able to investigate whether the alert was caused by an actual rogue switch or if it is a just a false alarm. This minimizes potential disruptions to the network. The effectiveness of this solution does not depend on the size of the network.

The drawback of this solution is that it takes a significant amount of time and effort to find the physical location of the rogue switch once the alert is sent (and determined to be a legitimate issue). The switch will remain in place until a network administrator can spend the time to investigate, locate the switch, and remove it. This leaves the network potentially vulnerable during that time. While much of this solution can be automated, some steps will require human interaction, leaving open the possibility of mistakes caused by human error.

A business using this solution would experience minimal to no interruption in day-to-day tasks. Unlike the DHCP solution, this solution does not cause network interruptions. Each switch would monitor the defined OIDs in addition to forwarding packets as normal. This solution does create extra tasks for the IT administrator, such as analyzing the trap messages to determine whether an alert leads to a rogue switch or if it was a false alarm. IT administrators would also have the responsibility of finding the physical location of the switch and removing it from the network.

 ISBN: 978-1-61208-729-0

C. MAC Address Whitelisting

This solution requires every switch to download a copy of a whitelist (maintained by the network administrator), which includes every MAC address that has a permission to transmit traffic on the network. As a switch receives a packet, it first checks to see if the sender MAC address is on the forwarding table. If there is no entry for the sender's MAC address, the switch references the whitelist. The MAC address is only added to the forwarding table if it is found on the whitelist; otherwise, the packet is flagged as potentially from a rogue device.

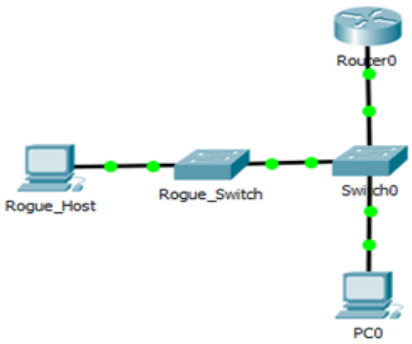

Fig. 6. Whitelisting example.

In Figure 6, Switch0, PC0 and Router0 are valid network devices. When Rogue_Host transmits a packet, Rogue_Switch will forward the packet to Switch0. Upon receiving the packet from Rogue_Switch, Switch0 will reference the whitelist, and will determine that Rogue_Host MAC address is not included. Therefore, Switch0 will drop the packet (or otherwise notify the network administrator of a potential rogue device on the network). It is possible to connect a rogue PC directly but it will still generate ARP broadcast queries, which is what we use for detection.

Given the benefits and drawbacks of this solution, whitelisting would be most ideal for smaller networks or for networks where the frame transmission speed is not critical. Smaller networks would allow for the network administrator to most effectively maintain the whitelist, and would also minimize the time that each switch takes to search the list for the sender's MAC address.

This solution is intended to detect and block rogue PCs (or other communicating devices), as well as protect the network against rogue switches added without permission. This solution does not provide information for the physical location of the rogue switch, though instead provides a layer of security to be used in parallel with other methods. This solution will always be successful in immediately detecting a rogue switch as soon as it begins to transmit data on the network, assuming that the MAC address of the sender is not *spoofed*. Only permitted devices will be able to communicate on the network, and all others will be blocked. The packets being transmitted would not even reach the destination before the rogue switch is detected, which allows for the packet to be intercepted by the IT administrator or dropped by the switch. No additional traffic is generated on the network as a result of the implementation of this solution apart from downloading the whitelist. Whitelisting is extremely effective in protecting the network. Detection of the rogue device using this method is almost immediate after the device begins to communicate, and 100% of the packets from the rogue device would be blocked from being delivered to their intended destination.

Whitelisting does not protect against an attack where the attacker spoofs the MAC address of the rogue device. In order to successfully spoof the MAC address, the attacker would either need in-depth knowledge about the addresses on the whitelist, or would need to repeatedly send frames with different MAC addresses until a valid address is discovered. Whitelisting also requires high maintenance from the IT administrator. Whenever a new device needs to be added (or an old device removed) from the whitelist, the list needs to be updated and a fresh copy is downloaded locally by each switch. This presents a vulnerability where a previously allowed device that has been removed from the whitelist may still be permitted to transmit on the network if the switch has not yet received an updated copy of the whitelist.

This solution is costly in terms of processing time and resources. Depending on the length of the whitelist, the packet forwarding time can also be impacted. The longer a whitelist, the longer time it takes to search through the list to determine if the sender is permitted to transmit on the network. By adding an extra step to a switch's forwarding process, the processing time increases linearly as the size of the whitelist increases.

Figure 7 compares the estimated maximum search time for whitelists of various sizes. The range selected represents a realistic whitelist size for small to medium sized networks. Companies with around 500-800 employees could have around 1,000 devices on the whitelist. If the time a switch takes to search the whitelist for a specified MAC address is around 3 microseconds per entry, then a relatively small whitelist with 10 entries would use up to 30 microseconds in the worst case (assuming linear search). A larger whitelist with around 1,000 entries would take up to 3,000 microseconds (0.003 seconds) to search. This greatly impacts the time from when the packet is initially sent from the source until it is received by the target.

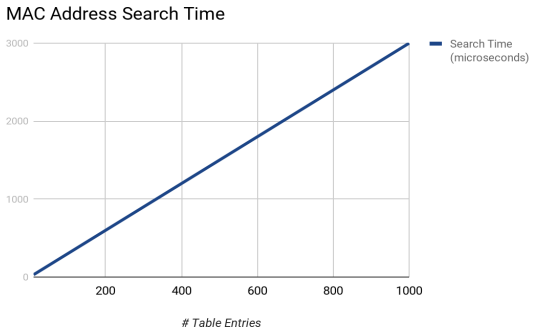

Fig. 7. Estimated Whitelist search time (based on linear search algorithm).

The larger the whitelist, the longer the potential search time for each switch. A whitelist with 100 entries would take

significantly less time to search than a whitelist with 100,000 entries.

Additional time may be spent sorting through false positives (or permitted MAC addresses that are initially incorrectly flagged as potentially rogue devices). This solution is designed to prefer false positives as opposed to false negatives (devices not permitted on the network that are incorrectly allowed network access), however, this is at the expense of processing time and inconvenience to the device attempting to access the network. False positives may be caused by a switch not having the latest copy of the whitelist. If a device is added to the whitelist after the switch updates its local copy, any traffic sent by the device will be blocked until the switch updates its copy again to include the new device.

MAC addresses can be easily spoofed to an address that is on the whitelist. If the whitelist addresses are not known to the attacker, a larger network would be easier for an attacker to randomly guess an allowable address. Based on these drawbacks, this solution would be ideal for smaller to medium networks.

V. CONCLUSIONS

In this paper, we proposed three solutions to detect a rogue switch. All three solutions can be implemented independently of each other, and all three solutions can be implemented at the same time. Each solution uses different sets of anomalies to detect a rogue switch. None of the three solutions rely on each other in order to be successful, which strengthens the network exponentially if all three solutions are implemented at the same time.

The whitelisting-based solution prevents devices that are not permitted from communicating on the network altogether. The DHCP-based detection solution assists in first finding if a rogue device exists, and second, finding the location of the device. The SNMP-based detection solution monitors the type of traffic being transmitted to determine whether a rogue device exists.

All three solutions will also work on any type of network, though each one has a specific network type where its maximum benefit can be reached. The whitelisting-based detection solution works best on smaller to medium size networks. The SNMP-based detection solution works best on networks that do not frequently have new devices connected. The DHCP-based detection solution works best on smaller to medium size networks where some downtime for hosts is acceptable.

To the best of our knowledge, this is the first such attempt to detect a rogue Ethernet switch (regardless of the vendor). We prove the feasibility of our solutions by carrying out realistic simulations and in-depth analysis. False positives are very unlikely because of the way our solutions are designed.

ACKNOWLEDGEMENTS

We would like to thank Adrian Mora for his initial contributions.

REFERENCES

[1] M. Rogers and D. Ruppersberger, “Investigative Report on the U.S. National Security Issues Posed by Chinese Telecommunications Companies Huawei and ZTE”, U.S. House of Representatives 112thCongress, 2012.

[2] “Cisco Adaptive Security Appliance Remote Code Execution and Denial of Service Vulnerability.”, Available at https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20180129-asa, 2018.

[3] P. Congdon *et al*, “IEEE 802.1X Remote Authentication Dial In User Service (RADIUS)”, Available at https://tools.ietf.org/html/draft-congdon-radius-8021x-10, 2001.

[4] 802.1AB-REV - Station and Media Access Control Connectivity Discovery, Available at http://www.ieee802.org/1/pages/802.1AB-rev.html, 2018.

[5] “NetSpot, FREE WiFi Site Survey Software for MAC OS X & Windows”, Available at: https://www.netspotapp.com/, 2018.

[6] “Nmap: the Network Mapper”, Available at https://nmap.org/, 2018.

[7] J. Kurose and K. Ross, “Computer Networking: A Top-Down Approach”, 6th ed. Upper Saddle River, New Jersey: Pearson Education, Inc., pp.27, 353-355, 480-482, 2013.

[8] “Configuring Spanning Tree Protocol”, Available at: https://www.cisco.com/c/en/us/td/docs/wireless/access_point/1300/12-2_15_JA/configuration/guide/o13span.html, 2018

[9] “Cisco Packet Tracer”, Available at https://www.netacad.com/courses/packet-tracer, 2018.

[10] ifInNUcastPkts(12), Available at http://oid-info.com/get/1.3.6.1.2.1.2.2.1.12, 2018.

[11] ifOutUcastPkts(17), Available at: http://oid-info.com/get/1.3.6.1.2.1.2.2.1.17, 2018

 ISBN: 978-1-61208-729-0

Towards a Multidimensional Model for Terrorist Attacks Analysis and Mining

Firas Saidi
RIADI Laboratory
National School of Computer Sciences, Manouba,Tunisia
e-mail: firas.saidi@ensi.rnu.tn

Zouheir Trabelsi
College of Information Technology, UAE University, Abu Dhabi, UAE
e-mail: Trabelsi@uaeu.ac.ae

Henda Ben Ghezala
RIADI Laboratory
National School of Computer Sciences, Manouba,Tunisia
e-mail:henda.benghezala@riadi.rnu.tn

Abstract— **The number of terrorist attacks have grown steadily in recent times, causing dire human and material damages. Analyzing potential terrorism data is without doubt an indispensable task, not only to understand these terrorist events, the terrorist implicated actors, their communities and their operation methods and tactics, but also to predict future attacks. However, terrorist events are commonly conducted differently and rarely fit the models of conventional attacks. Thus, terrorist events are usually hard to analyze, mine and investigate. In this paper, we propose a multidimensional model, named Terrorist Data Warehouse (TerDW) for allowing terrorist investigators to perform interesting and specific analysis for decision-making objectives. Practically, while consulting gathered information relative to specific terrorist attacks, such as attacks' locations, dates, used weapons, and implicated attackers, investigators can query the proposed data warehouse (TerDW) using multidimensional queries. The proposed model is based on consistent dimensions and measures. Terrorist attacks data are extracted from the Global Terrorism Database (GTD), a well-known database that includes information about terrorist events that took place from 1970 to 2017. Queries examples have been conducted to evaluate the efficiency of the proposed model. The queries' results demonstrate clearly that the proposed model allows investigators to carry out more appropriate multidimensional analysis, investigations and decisions.**

Keywords- Terrorist Attack; GTD; TerDW; MDX; OLAP analysis.

I. INTRODUCTION

Terrorist organizations try to multiply their malicious activities and to change their tactics, especially with the emergence of Web accessible devices. In fact, investigators are trying to gather useful information about these groups to devise a prevention strategy. Many researchers on terrorism constructed very interesting data sets about terrorist incidents around the word such as: Global Terrorism Data Base, John Jay and ARTIS Transnational Terrorism Database (JJATT), and Chicago Project on Security and Terrorism [1].

Thus, analyzing potential data related to terrorist events is without doubt an indispensable task, not only to understand the terrorism phenomenon, terrorist implicated actors, their operation methods and tactics, but also to predict and prevent future attacks.

However, terrorist events are commonly conducted differently and rarely fit the models of conventional attacks. Thus, terrorist events are hard to analyze, mine and investigate. To overcome this issue, we propose a multidimensional model, named Terrorist Data Warehouse (TerDW), for allowing terrorist investigators to perform interesting and specific analysis to help their decision-making. Practically, data gathered about terrorist attacks and incidents such as attacks' locations, dates, used weapons, and implicated actors are interesting to be visualized as multidimensional queries' results for decision- making goals. TerDW model includes consistent dimensions and indicators to analyze terrorist data extracted from the GTD database [2]. Interesting queries examples performed using TerDW are discussed in this work.

This paper is organized as follows: Section 2 describes related works on terrorist groups and activities identification and analysis. Section 3 presents and discusses the proposed framework architecture. In Section 4, the multidimensional model TerDW is discussed in details. Section 5 discusses how TerDW can be used for OnLine Analytical Process (OLAP) analysis and presents results of queries examples. Section 6 discusses the advantages and limitation of the proposed model. Section 7 concludes the paper and outlines perspectives for future research works.

II. RELATED WORKS

Various research works focused on the detection, analysis and mining of terrorist organizations and their malicious activities. These works are categorized into two approaches, namely Social Networks Analysis (SNA)-based approaches and hybrid approaches [1].

In [3], the SNA methodology was applied on the Jemaah Islamiyah cell that was responsible for the Bali bombings in 2002. Furthermore, an intelligent analysis framework was proposed to understand the structure of such a cell and to

 ISBN: 978-1-61208-729-0

predict with real time analysis the outcomes of these terrorist subgroups.

In [4], the authors studied Al Qaeda network characteristics and activities. In fact, the authors argued that Ben-Laden occupied a central position within the Al Qaeda network based on the computation of some measures.

The work in [5] presented a list of techniques that may analyze new summaries of terrorist attacks from the GTD database. The authors emphasized the use of text mining to extract critical concepts from free text. In a second phase, they used different learning algorithms including Naive Bayes, decision tree and support vector machine, to learn the terrorism incident, and especially, to detect its types from the news.

In [6], the authors proposed a dimensional model used in analyzing cybercrime data. Furthermore, they presented some interesting queries and the types of cybercrime analysis that can be performed based on the proposed model.

In [7], the authors introduced a theoretical framework which has two aims: first, to discover criminal organizations in networks issued from phone calls records and, second, to help investigators to analyze their structure and to discover hidden roles and relationships. Thus, this work contributes to terrorist network visualization from mobile phone calls and to community detection and analysis.

In [8], the authors introduced a technique for terrorist network destabilization based on two main steps, namely, first, communities detection and, second, the analysis of these groups by the identification of key actors and potential links.

III. TERRORIST DATA WAREHOUSE (TERDW) MODEL

In order to analyze terrorist attacks and incidents, we use a multidimensional model of data warehouse. Practically, we describe the methodology used to design the TerDW model. In fact, we refer to the well-known methodology of Kimball [9] to design the proposed Data Warehouse. Indeed, this process follows four main steps, as shown in Table I.

A fact table operates with dimensions and holds the data to be analyzed. A dimension table stores data about the ways in which the data in the fact table can be analyzed. Furthermore, we note that dimensions and measures of the multidimensional model are inspired from the GTD data base structure [2].

Figure 1 describes the proposed TerDW model, which is a galaxy model with two fact tables, namely Terrorist incident and Terrorist attack tables. Practically, the Terrorist attack table is used to analyze attacks, while the Terrorist incident table is used to analyze crimes at each incidence level.

We note that the **Galaxy schema** (Figure 1) is a Data Warehouse model, which contains two or more fact tables sharing some dimension tables. Sharing dimension tables can reduce database size, especially where shared dimensions have many possible values.

The proposed model can be exploited for decision making by investigators and terrorism specialists.

We identify two facts, namely, **Terrorist Incident fact** and **Terrorist Attack fact,** used to analyze terrorist events following different axis, i.e., dimensions and using different computed values, i.e., measures.

TABLE I. KIMBALL METHODOLOGY FOR TERDW MODEL IMPLEMENTATION

#	Steps	Description
1	Select the business process	Our business process is terrorist attack investigation.
2	Declare the grain, which establishes exactly what a single fact table row represents	In the Global Terrorism Database (GTD) [2], there are two types of terrorist attacks, namely: incident and attack. The incidence is a suspicious activity that may or may not result in a terrorist attack. **Facts:** Terrorist Incident (F1) and Terrorist Attack (F2).
3	Identify the dimensions used to analyze the fact table	**Dimensions:** Date, Attacker, Perpetrator organization, Location, Incident information, Attack information, Weapon, Target (victim), Demographic and Social information.
4	Identify measures of the fact table	Measures: Total number of fatalities, Number of Perpetrator Fatalities, Total Number of Injured, Number of Perpetrators Injured, Value of Property Damage (in USD), Total Number of Hostages/ Kidnapping Victims, Hours of Kidnapping / Hostage Incident, Days of Kidnapping / Hostage Incident, Total Ransom Amount Demanded, Total Ransom Amount Paid, Number Released/Escaped/Rescued

 ISBN: 978-1-61208-729-0

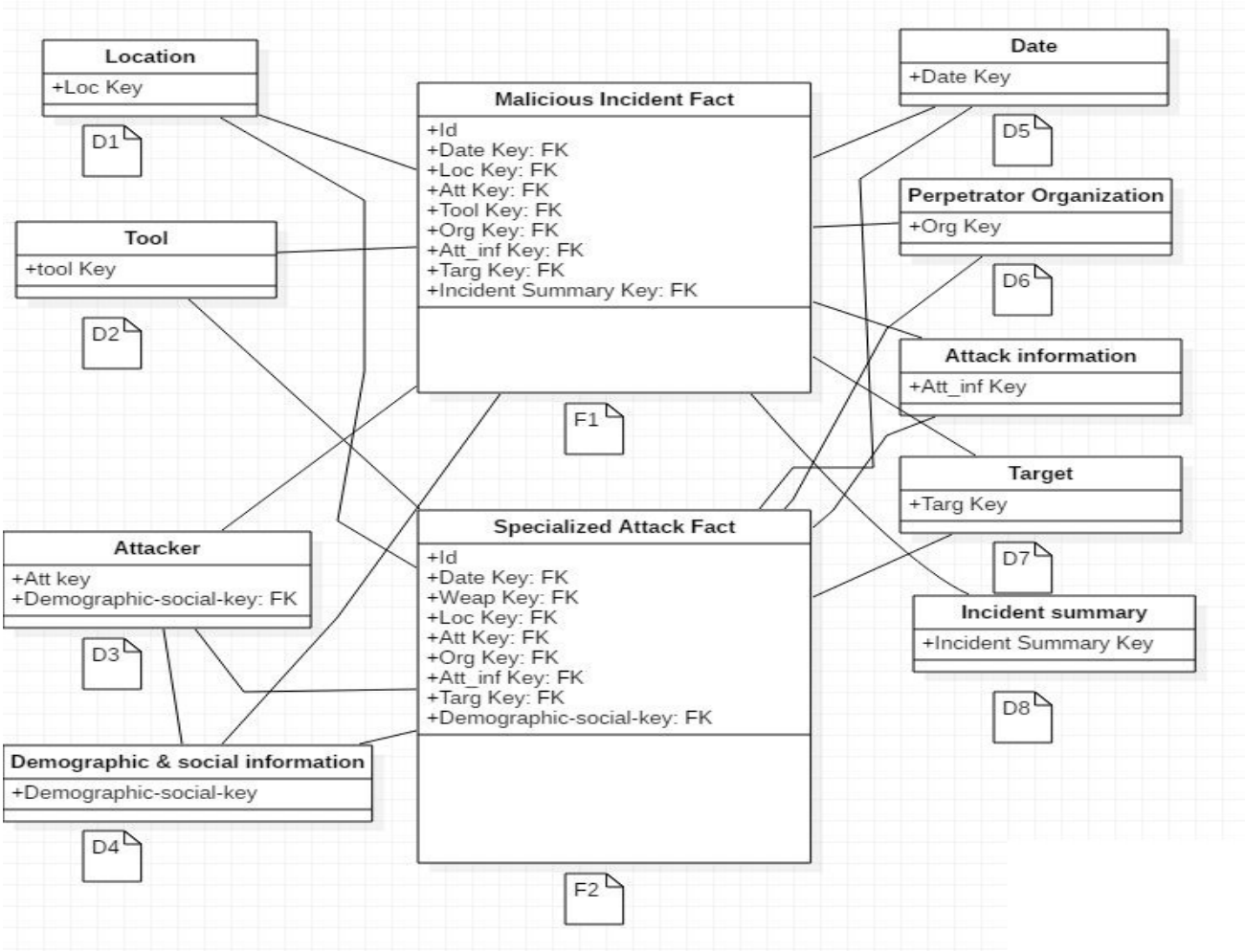

Figure 1. Galaxy Terrorist Data Warehouse (TerDW) schema

IV. TERRORIST ATTACKS ANALYSIS PROCESS

In this section, we present the process of terrorist attacks analysis in Figure 2. We apply the Extract Transform Load process on GTD database. Data are loaded on a TerDW, which is used to analyze and mine terrorist attacks based on various dimensions.

Investigators can ask the OnLine Analytical Processing (OLAP) cubes [10] constructed from the TerDW and can perform multidimensional analysis or apply data mining techniques in order to extract valuable knowledge about terrorist attacks and their related dimensions. Practically, the data warehouse is powered by data from GTD data base [2]. Indeed, data are extracted, transformed and loaded using an open source as Extract Transform Load (ETL) tool.

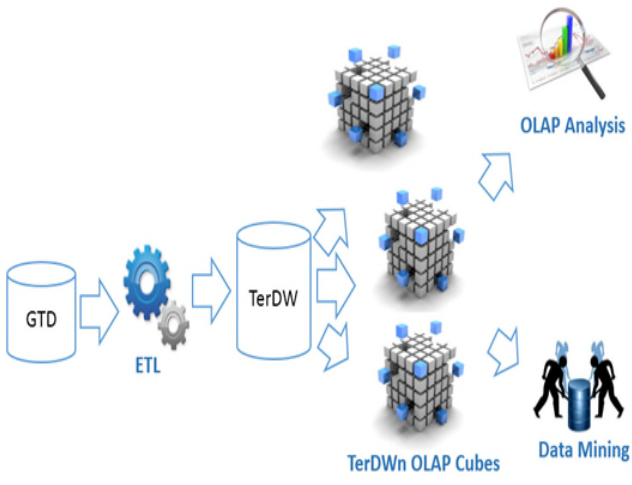

Figure 2. Terrorist attacks analysis process

V. OLAP ANALYSIS OF TERRORIST ATACKS

In this section, we present some multidimensional analysis based on TerDW to help investigators in predicting terrorist attacks and destabilizing terrorist cells (see Figure 3).

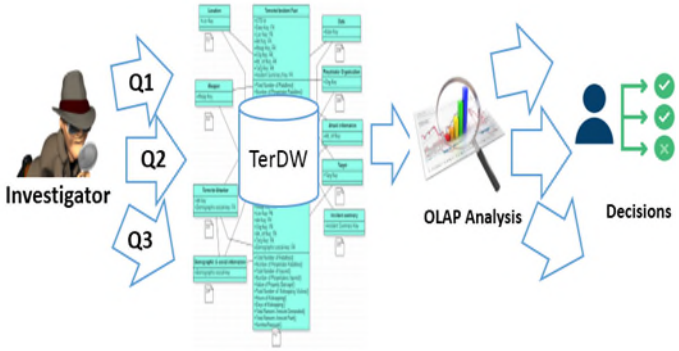

Figure 3. Investigator's decision making process

Based on the dimensions and measures defined in Figure 1, many interesting Multidimensional Expressions (MDX) (queries) can be executed to analyze data of terrorist attacks data. Figure 4 presents an example of MDX query which aims to extract the total number of injured and rescued (Total Number and Rescued Number) of 2011 and 2016 attacks (Date dimension) in Africa (Location dimension) from TerDW-1 OLAP cube (includes Terrorist Attack fact).

```
SELECT
   { [Measures].[Total Number of
Injured], [Measures].[Number Rescued] },
{[Date].[Attack
information].[Year].&[2011],[Date].[Att
ack information].[Year].&[2016],}     ON
1
   FROM [TerDW1]
   WHERE ([Location].[Africa])
```

Figure 4. Example of MDX query

The TerDW model allows investigators to submit interesting queries on specific terrorist events and actors (Figure 5).

Investigators can have an idea about establishments and targeted persons, most frequent attacks, used weapons, periods of frequent attacks, demographics and social information of attackers, information about terrorist organizations and can quantify different human and material damages.

Figures 6, 7 and 8 present the results of three queries applied on the proposed TerDW data model. Figure 6 shows the most frequent terrorist attack's type that took place between 1995 and 2012. The most frequent attacks are bombing and explosion, armed assault and infrastructure attacks respectively.

Figure 7 shows statistics about the most targeted victims during the terrorist attacks in the GTD database. Military, police and business actors are the most targeted in terrorist attacks.

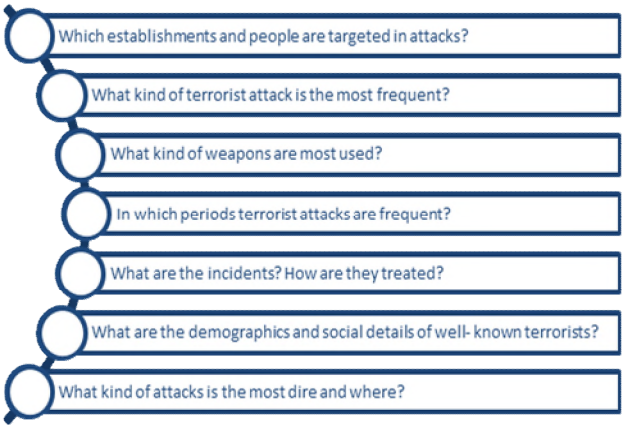

Figure 5. Queries samples for OLAP analysis

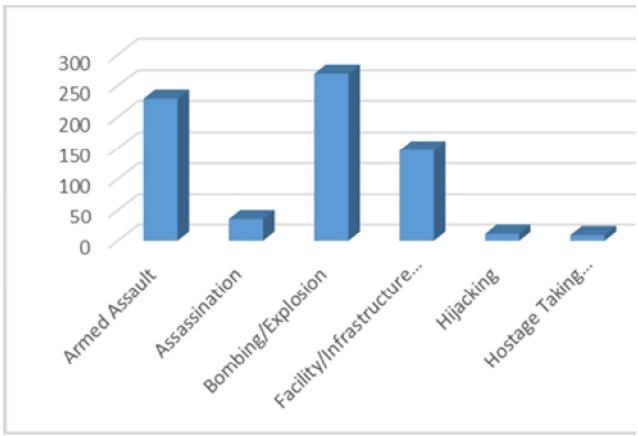

Figure 6. Terrorist attack type frequency between 1995-2012

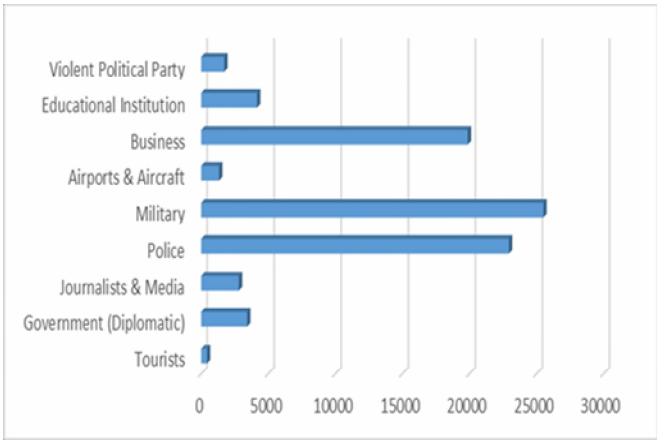

Figure 7. Most targeted victims in terrorist attacks

Figure 8 shows the percentages of the most used weapons during terrorist events in the GTD database. Firearms, explosive/bombs and dynamite and incendiary devices are the most used weapons in terrorist attacks, respectively.

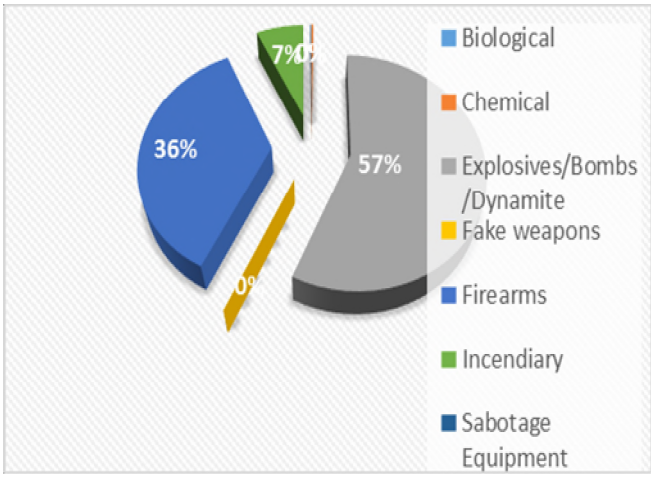

Figure 8. Most used terrorist weapons

Thus, investigators can extract data and statistics about terrorist attacks, used attack tools and techniques and the attack actors. In fact, terrorists might be key actors of cells or criminal organizations and might not belong to any terrorist community. This information is important to identify the well-known ‘’lone wolves” actors [11]. A “lone wolf” actor is a terrorist who prepares and executes terrorist acts alone, i.e., outside of any command organization and without the assistance of other terrorist actors. In fact, this type of terrorists is very dangerous since it is difficult to identify and to predict.

Implementation and Evaluation

In order to experiment the proposed multidimensional model, we used SQL Server BI framework. We created TerDW following the conceptual model described in the paper. After that, we used the SQL Server Integration Services (SSIS) as Extract Transform Load (ETL) tool. From the GTD database, data were extracted, transformed and loaded to the TerDW. For OLAP analysis, we used SQL Server Analysis Services (SSAS) tool.

Investigators can create various interesting reports in an easy manner using SQL Server Reporting Services (SSRS).

VI. DISCUSSION

The proposed Data Warehouse TerDW is a multidimensional model used for terrorist attacks and incidents analysis. Practically, investigators can query the TerDW to answer various interesting questions about attacks, attackers and used techniques. Indeed, this pack of information can be indispensable to understand terrorist organizations following different axes, i.e., dimensions, their topology, tactics and operation methods.

However, this model is limited and cannot provide deep information and correlations using simple MDX queries. Hidden relationships between data can provide interesting knowledge. Thus, TerDW must be valorized by data mining techniques in order to extract valuable knowledge and to discover new information not only used to understand terrorist events, but also to predict future attacks.

VII. CONCLUSION

In this paper, we proposed a data warehouse (TerDW) which is a multidimensional model for decision-making. We explained how TerDW can be used for OLAP analysis to help investigators in their work not only to understand the terrorism phenomenon, but also to predict others related criminal events. Furthermore, we detailed how this model can answer some interesting questions and we presented different results of the aforementioned OLAP queries. As an interesting perspective of this work, data mining techniques can be used to extract valuable knowledge about terrorist events and actors from TerDW. Moreover, a machine learning process can be applied on GTD to predict future attacks.

REFERENCES

[1] F. Saidi, Z. Trabelsi, K. Salah and H. B. Ghezala. ”Approaches to analyze cyber terrorist communities: Survey and challenges”. Computers Security, vol. 66, pp. 66-80, 2017.

[2] G. LaFree and L. Dugan, “Introducing the global terrorism database”, Terrorism and Political Violence, vol. 19, no. 2, pp. 181-204, 2007.

[3] S. Koschade, “A social network analysis of Jemaah Islamiyah: The applications to counterterrorism and intelligence”, Studies in Conflict & Terrorism, vol. 29, no. 6, pp. 559-575, 2006

[4] E. Wu, R. Carleton, and G. Davies, "Discovering bin- Laden’s replacement in al-Qaeda, using social network analysis: a methodological investigation”, Perspectives on Terrorism, vol. 8, no. 1, pp. 57-73, 2014.

[5] S. Nizamani and N. Memon, “Analyzing News Summaries for Identification of Terrorism Incident Type”. Educational Research International, vol. 3, no. 4, 2014

[6] I. Y. Song, J. D. Maguire, K. J. Lee, N. Choi, X. Hu and P. Chen, “Designing a data warehouse for cyber crimes”, Journal of Digital Forensics, Security and Law, vol. 1, no. 3, pp. 1, 2006.

[7] J. J. Xu and H. Chen. ”Fighting organized crimes: using shortest-path algorithms to identify associations in criminal networks”. Decision Support Systems, vol. 38, no. 3, pp.473-487, 2004.

[8] D. Anggraini, S. Madenda, E. P. Wibowo and L. Boumedjout, “Network Disintegration in Criminal Network”, In 11th International Conference on Signal-Image Technology Internet-Based Systems (SITIS), pp. 192-199, 2015.

[9] R. Kimball and M. Ross, “The data warehouse toolkit: the complete guide to dimensional modeling”, John Wiley & Sons. 2011.

[10] T. Niemi, M. Niinimäki, J. Nummenmaa and P. Thanisch, “Constructing an OLAP cube from distributed XML data”, In Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP , pp. 22-27, 2002.

[11] P. J . Phillips, “Lone wolf terrorism”, Peace Economics, Peace Science and Public Policy, vol. 17, no. 1, 2011.

 ISBN: 978-1-61208-729-0

A Large-Scale Analysis of Browser Fingerprinting via Chrome Instrumentation

Mohammadreza Ashouri
Institute of Computer Science - University of Potsdam
Potsdam, Germany
email: ashouri@uni-potsdam.de

***Abstract*—In this work, we introduce FPTracker as a standalone, portable and practical browser that utilizes static and dynamic analysis to obtain concise results on a large set of websites. In contrast to the previous works, which rely on native code instrumentation that have low performance and high cost for monitoring each fingerprint Application programming interface (API), FPTracker is developed as an independent tool that does not need to interact with users' web browsers. In order to prove the usefulness of FPTracker, we have evaluated the top 10,000 European websites (according to Alexa.com) that comprise 1,393,426 links. We have chosen popular European websites to discern how these websites employ user tracking third parties concerning the EU General Data Protection Regulation (GDPR). Accordingly, we found that 117,012 links out of 1,393,426 use invisible user fingerprinting systems. For instance, one of the biggest European banks and a leading advertising website still fingerprint their visitors.**

Keywords - browser fingerprinting; ***privacy***; ***security***; ***web API***; ***encryption.***

I. INTRODUCTION

Browser fingerprinting is a user tracking technique which extracts a wide range of unique information about online users through web browser APIs. Numerous methodologies have been proposed in order to dispute with browser fingerprinting, often are based on static analysis techniques and predefined blacklists. However, despite recent advances in privacy enhancement in web browsers, fingerprinting systems are not only improving their accuracy but also trying to avoid being caught by anti-tracking systems.

The information obtained in this technique includes web browser software and version (e.g., Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge, Opera), operating system (Windows, Mac OS, Linux), screen resolution (mobile, tablet or desktop), established fonts, browser plugins and extensions, time zone, ad-blockers, language, and the hardware properties of users device.

These items may not be individually identifiable; for instance, millions of users may use the same version of Chrome browser. However, the combination of these pieces of information is enough to identify users uniquely. In other words, there is enough distinction among the aforementioned features so that only one in several thousand can have the same combination of blueprints, which is perfectly accurate for the fingerprinting objectives (e.g. advertising). Hence, a successful web fingerprinter relies on this slight variance between users devices in order to create and label online users by unique hash strings (unique fingerprint IDs).

When for the first time, Electronic Frontier Foundation (EFF) published research called "How Unique is Your Browser?" in May 2010, browser fingerprinting got widespread attention. Via analyzing over a million visits to their research website [36], they found that 83.6% of the browsers seen had a unique fingerprint; 94.2% between those with Java or Flash enabled. It also proved that the combination of its fingerprint algorithm had at least 18.1 bits of entropy, meaning that users had a 1 in 286,777 chance of receiving the same fingerprint hash string as another one. The interesting point is that they got this fingerprints only through 8 web browser features (e.g., fonts, plugins, timezone, supercookies, cookies enabled, user agent, Hypertext Transfer Protocol (HTTP) accept, screen resolution). However, due to the advancement in web browser software and privacy systems, some of these features are out of function (e.g. Flash). Therefore, it sounds that by elaborating of functional features of new browsers, it is possible to extract accurate browser fingerprint of the users which is a desirable target for advertising companies.

Moreover, browser fingerprinting is on a debated subject with privacy laws. Compared to more well-known tracking cookies, browser fingerprinting is complicated for users and browser extensions to contend: websites can do it without disclosure, and it is difficult to modify browsers. As cookies are more noticeable and more comfortable to tackle, corporations have been more moved to turn to trickier fingerprinting techniques. Companies also have to obey the law as well as the residents of the European Union. However, during this study, we found that still many European websites including online banks still use this technique regardless of the law.

Web browser fingerprinting methods have been changing based on the new features providing by the full-fledged web browsers as well as the recent development in anti-tracing and anti-fingerprinting systems (e.g. ad-blockers). Hence, in this work we introduce FPTracker, which analyzes the presence of advanced and encrypted fingerprinting scripts and third-parties in websites. FPTracker can also enhance the privacy level of web users via identifying and reporting the latest fingerprinting tricks on the web.

Accordingly, we built FPTracker based on similar technologies used in previous works, such as FP-STALKER [26], FPGaurd [18], and FPDetective [20]. However, it has several key differences that provide more accurate, thorough, and

 ISBN: 978-1-61208-729-0

reliable analysis. Hence, the key contributions in our work are as follows:

1) **Minimal cost of monitoring new fingerprinting APIs**. While most of the previous works rely on native instrumentation code [20]–[22] (produce a high maintenance cost and a high cost-per-API monitored), in FPTracker the monitoring cost of new fingerprinting APIs is minimal. It means that fingerprinting metrics can be enabled or disabled by users without web browsers recompilation or rendering engines manipulation. This feature also permits users to conduct their customized analysis.
2) **Performing in-depth analysis**. FPTracker uses a a new combined method based on static and runtime analysis, which enables us to perform more in-depth analysis and get accurate results.
3) **Analyzing encrypted scripts**. Since the industrial fingerprinting libraries tries to utilize encryption methods to evade anti-fingerprint systems (e.g. ad-blockers), FPTracker analyzes encrypted scripts to recognize their actual intention via a light and innovative approach (this is explained in the next sections).
4) **Running independently**. Our approach works based on the instrumentation of Chrome driver [33]. Therefore, our proposed tool does not need to communicate through the installed user web browsers (e.g., Chrome, Firefox, Opera) and makes FPTracker standalone and agile in comparison with similar tools.
5) **Simulating user interactions**. In order to bypass anti-crawler mechanisms in industrial browser fingerprint third parties, FPTracker performs arbitrary user interactions on websites to deceive the anti-crawler systems and traces the hidden APIs that work by user interactions.
6) **Identifying new fingerprinting techniques**. In FPTracker we have specified a long set of distinct metrics almost based on the recent fingerprinting approaches (e.g., WebRTC, WebGL, etc.) . We also consider other common techniques, such as canvas fingerprinting and font enumeration. Hence, we have achieved concise and more reliable results in comparison with previous studies which only focus on one or a few metrics.
7) **Analyzing secondary web pages**. Many popular web sites are interested in tracking their visitors based on their search results or favorite items (which are typically located on secondary pages, such as searching or booking pages). Thus, unlike previous studies that only analyze homepages, FPTracker detects the presence of web tracking scripts in all website pages including homepages and secondary pages.

In the following section, we introduce Browser fingerprinting techniques. Next, in the methodology section, we present our approach, and in the implementation section, we describe the details of FPTracker implementation. In the experimental results section, we express our evaluation results. Finally, in the related works section, we will compare our proposed system with similar works, and we represent the advantages of FPTracker in comparison with the previous works.

II. BACKGROUND

Web-Based fingerprinting requires certain types of information which are collected from user's systems. A client's browser requests such information. Executed web scripts like JavaScript which are placed on different web pages determine the type of collected information. The operations of client-side fingerprinting scripts are done based on HTTP header fields. It is one of the primary data transfer mechanism in many popular browsers. Unparalleled retrievement of data from web servers is one of the critical features of client-side fingerprinting which gives the web trackers, such as advertisement third parties, the ability to authorise online users based on their environment properties directly.

The computed fingerprint is a unique identifier which stores collected data on specific databases. The contents of these databases are defined after the completion of necessary processes in the browser. Eventually, the acquired results are sent to the web server.

In more advanced web-based fingerprinting schemes, the website provider is replaced by a third-party agent. Fingerprinting script is anonymously inserted in the website's main code only under permission of owners or business partners. In other words, such web-scripts are invisibly retrieved by visitors. The new scheme provides a more comprehensive set of information about users, and they are referring to websites. This technique might be useful only in cases where the website owner agrees on publishing advertising content on different parts of web pages.

A. Browser behavior

Every browser (type and version) has its unique characteristics which lead to unprecedented behavior. Such differences can be used for the creation of unique fingerprints.

JavaScript can gather information about screen properties of user devices. "Windows. Screen" object is the unique tool allocated to this process. Characters like depth, width, height and colour of devices screen can be gathered by such object which leads to a determination of the type of device (smartphone, tablet, two screened PCs). Lack of generalisation is a severe problem for this technique since different browsers use different run-time codes which interrupts the comparability of fingerprints. Some features of users systems like the number of CPUs or available free RAM can be useful for browser fingerprinting. The extraction of such information is not possible for all browsers, but the rest of the collected data can have critical applications for advertisers and web-owners.

B. Font detection/ enumeration

Any information in the OS is shown using fonts. These visual symbols are considered as unique features of devices. Also, installed software like MS-word or Acrobat creative suite use their special fonts, and on their front, the installation of customized fonts by users increase the diversity of available

 ISBN: 978-1-61208-729-0

fonts in any system. The fingerprinting procedure might use these differences in order to attribute a unique signature to any device. The list of enumerated fonts is a meaningful basis for collecting a significant amount of data about the system's situation. The more comprehensive the font database, the higher the accuracy of fingerprinting based on font features.

C. Canvas fingerprinting

Canvas is an HTML5 API which is used to draw graphics and animations on a web page via scripting in JavaScript. However, apart from this, Canvas can be used as further entropy in web-browser's fingerprinting and applied for online fingerprinting targets. The approach is based on the fact that the corresponding canvas image may be rendered uniquely in distinct machines. This occurs for numerous causes. At the image format level, web browsers use diverse image processing engines and compression level; therefore, the final images can get distinctive checksum [25]. At the system level, operating systems have many fonts which use different algorithms for pixel rendering. Listing 1 represents an example of Canvas fingerprinting in JavaScript.

Listing 1. Sample code showing how Canvas fingerprinting works

```
function doFingerprinting(evt) {
    let canv = document.createElement("CANVAS");
    canv.height = canv.width = windSize;
    canv.style.height = canv.style.width = windSize + "em";
    let ctx = canv.getContext("2d");
    ctx.fillStyle = "black";

    let d = document.createElement("DIV");
    d.style.position = "fixed";
    d.style.left = d.style.top = "0px";
    d.style.zIndex = -1000;
    d.style.visibility = "hidden";
    document.body.appendChild(d);
...
```

D. WebGL fingerprinting

WebGL is a JavaScript API for rendering interactive graphics within any web browser without using extra plugins. Since GPU makes the rendering process of interactive graphics, the effect can be different in machines with different GPUs [24].

E. WebRTC tracking

WebRTC is an API with a complex set of protocols for establishing communications applications. It is created for applications such as voice and video chat (e.g., Facebook Messenger, OpenTokRTC, Google Hangouts) [32]. WebRTC offers TCP-like reliable and UDP-like unreliable data channels. Since WebRTC has recently become generally accessible and well established in web browsers, this API has become attractive for user fingerprinting purposes.

III. METHODOLOGY

Regarding our studies, almost all previous studies detect browser fingerprinting scripts in web pages based on web crawling and performing static analysis. This traditional approach is losing its functionality due to the recent advancements in fingerprinting techniques (such as using anti-crawling, anti-robot and code encryption methods to track real users instead of web bots). Moreover, we have observed that the popular browser fingerprinting libraries (e.g., Bluecava, fingerprint.min.js, client.js, google analytics, etc.) only fingerprint the actual users. In other words, these scripts do not run their fingerprinting functions until online users interact on web pages.

In this work, we attempt to support all of the practical features in FPTracker. Hence, we have created a static analyzer that co-operates with a runtime analyzer. We have also specified our fingerprint metrics based on the new techniques and recent updates in the popular user tracking third parties.

In this section, we explain the outline of our method, and in the next section, we present the details of the implementation.

Figure 1 represents an overview of FPTracker, a standalone web browser inside FPTracker is responsible for loading websites, enabling the analyzers to work on the interactive web pages and performing user interaction on the loaded web pages. After loading a website, our investigation will be conducted based on the following steps:

1) The static analyzer starts as a background thread in order to find any match between the enabled fingerprint APIs (Fingerprint Metrics) and the web page loaded objects (e.g., DOM, JavaScript, and CSS).
2) The runtime analyzer performs user interactions on the loaded web pages to trigger and trace the potential **hidden fingerprinting functions**.

A. Static Analyzer

When a webpage is entirely loaded on the internal browser, the Static Analyzer begins to parses the webpage as a set of DOM objects and separates the JavaScript, CSS, and HTML scripts. Then, the analyzer looks for any match between the tool enabled fingerprinting APIs and the extracted web APIs of the web page (Figure 2 shows the static analyzer in FPTracker).

We present some of the web metrics that we have specified for FPTracker in Table I. The purpose of the specification of these metrics is their usability in browser fingerprinting as well as their output values, which combines with each other to precisely identify online users. We obtained these metrics by investigating popular fingerprinting libraries and scripts. The Static Analyzer iterates this process to analyze all of the web pages and third party libraries of an examined website. After ending the process, FPTracker submits the identified fingerprints to the scoring module, which is responsible for providing the details of identified fingerprinting APIs in a website for the report database.

B. Runtime Analyzer

The Runtime Analyzer module is responsible for detecting and tracking suspicious JavaScript calls in websites, which

 ISBN: 978-1-61208-729-0

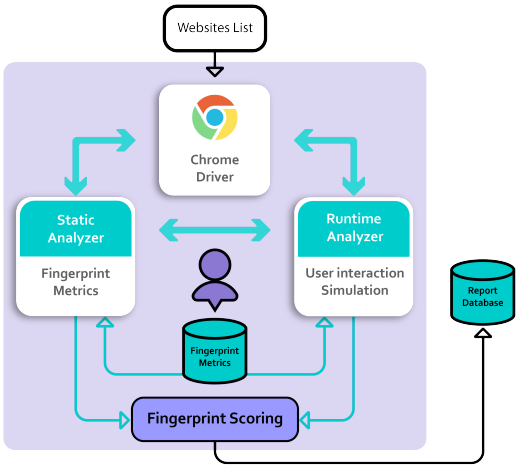

Figure. 1. Framework overview

TABLE I. SOME OF THE SPECIFIED JAVASCRIPT METRICS IN FPTracker WHICH CAN BE USED IN BROWSER FINGERPRINTING

Object Name	Usage
navigator.plugins	List of Plugins
navigator.userAgent	User agent header sent by the browser
navigator.language	Preferred language of the user
navigator.languages	Preferred languages of the user
navigator.cpuClass	CPU class of the user's OS
navigator.geolocation	Access to the location of device
navigator.platform	Platform of the browser
navigator.hardwareConcurrency	Number of logical processor cores available
navigator.deviceMemory	Device memory in gigabytes
navigator.doNotTrack	User do-not-track preference
navigator.appName	Official name of the browser
navigator.connection	Information about network connection
screen.colorDepth	Bit depth of color palette for displaying images
screen.height	Total height of users screen in pixel
screen.width	Total width of users screen in pixel
window.localStorage	Access a session storage object for the document origin
window.sessionStorage	Access a session storage object for the current origin
window.indexedDB	Store the data inside user browser
window.RTCPeerConnection	WebRTC connection
window.AudioContext	Audio-processing graph

either are encrypted or stimulated by user interactions. This module, which is shown in Figure 3, has two main threads:

The first thread is responsible for simulating user interactions on web pages (e.g., moving the mouse, clicking on buttons, etc.). In the meanwhile, the runtime analyzer traces hidden fingerprinting calls that serve for user interactions. For instance, in the case of **fingerprint.js**, which is a popular browser fingerprinting library, user uniques hash ID will be generated when an online user start to interacts on web pages (Listing 2 presents a sample hash function.).

Listing 2. Sample browser fingerprinting hash function

```
var my_hasher = new function(value, seed){ return value.length %
     seed; };
var fingerprint = new Fingerprint({hasher: my_hasher}).get();
```

The second thread of the **Runtime Analyzer** (RA) is responsible for intercepting and monitoring encrypted JavaScript codes inside of a web page. During our research, we have discerned that real-life user tracking third parties take advantage of web encryption algorithms to evade being caught and blocked by anti-tracking/ anti-fingerprinting systems (e.g. ad-blocking web extensions). Hence, it is inevitable that FPTracker must be able to disclose fingerprinting scripts even in encrypted formats. Therefore, the RA module analyzes the actual purposes of the encrypted scripts through monitoring web pages function calls, which is done via lightweight instrumentation of JavaScript engine. We describe the analysis of this technique in the next section.

IV. IMPLEMENTATION

In this section, we describe the implementation of FPTracker. As we mentioned earlier, our tool consists of three main modules: an internal instrumented browser, a static analyzer and a runtime analyzer. The entire platform is built on Python, **Selenium** [23], and **Chrome web driver** [33]. Selenium is a compact framework for inquiring web appli-

 ISBN: 978-1-61208-729-0

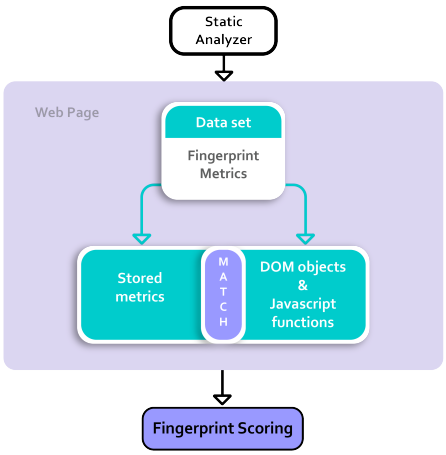

Figure. 2. Static Analyzer in FPTracker

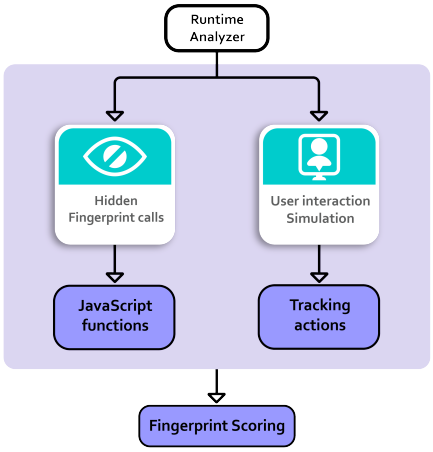

Figure. 3. Runtime Analyzer in FPTracker

cations. Selenium gives a playback tool for creating sound, browser-based regression automation tests. We also use **Beautifulsoup** [35] for parsing the contents of web pages. Beautiful Soup is a library that makes it simple to parse HTML and XML documents. Moreover, It produces a parse tree for parsed pages that can be used to obtain data from HTML, which is helpful for web scraping.

A. Internal Browser

The internal browser in FPTracker is responsible for presenting reality and support for Web APIs. We considered a variety of choices to drive measurements, i.e., to instruct the browser to visit a set of pages and to perform a set of actions on each. The two main categories to choose from are lightweight browsers like PhantomJS and full-fledged browsers like Chrome and Firefox. We decided to use Selenium which is a cross-platform web driver for Firefox, Chrome, and PhantomJS. By using our internal browser, all technologies that web users would have access to (e.g., HTML5 storage options, Canvas, WebGL, etc.) are also supported by FPTracker. The alternative, PhantomJS, does not support WebGL, HTML5 Audio and Video, CSS 3-D, and browser plugins making it challenging to run measurements on the use of these technologies. In retrospect, this has proved to be a sound choice. Without full support for new web technologies, we would not have been able to discover and measure the use of new fingerprinting techniques (such as webRTC and WebGL) or interacting on web pages.

In order to use Selenium, a real browser should be instrumented. Therefore, we have chosen Chrome Driver to be instrumented by Selenium because Chrome is the most popular web browser which takes 64% of the global browser market based on Browser Market Share Worldwide in December 2018 [34]. Hence, using the Chrome Driver make our internal web browser similar to an ordinary web user.

B. Static Analyzer Module

The Static Analyzer (SA) runs as a background thread, and it extracts web APIs of a given web page and looks for tracking scripts in the load web pages. All the specifications including fingerprint metrics are stored in a plain text file which is called **Fingerprint Metrics**. Users can update this file without recompilation of FPTracker or struggling with low-level data structures. Listing 3 shows how users can update and specify Fingerprint Metrics in a plain text file.

Listing 3. How users can update the fingerprint metrics in FPTracker

```
navigator.plugins : TRUE ;
navigator.hardwareConcurrency : FALSE ;
navigator.connection : TRUE ;
window.localStorage : FALSE ;
window.AudioContext : TRUE ;
```

The SA module examines the whole DOM objects of a given web page, and then it looks for any match between the stored metrics with parsed DOM objects and JavaScript functions. For instance, Figure 1 shows a Canvas API that is commonly used in fingerprinting libraries. In this sample, Listing 4 presents how the SA module detects canvas fingerprint functions in a given web page.

Listing 4. Sample code showing how FPTracker identifies Canvas fingerprinting

```
{
 var methods = []
 var canvasMethods = ['getImageData', 'getLineDash',
      'measureText','getContext','isPointInPath']
 canvasMethods.forEach(function (method) {
   var item = {
     type: 'Canvas',
     objName: 'CanvasRenderingContext2D',
     propName: method
   }

   methods.push(item)
 })
```

C. Runtime Analyzer Module

As we mentioned earlier, the Runtime Analyzer (RA) has two goals, simulating user interactions on websites and analyzing encrypted scripts inside of web pages.

Listing 5. Sample code showing how CrytoJS encrypt scripts in web pages

```
var encrypted = CryptoJS.AES.encrypt("Message", "Secret
     Passphrase");
var decrypted = CryptoJS.AES.decrypt(encrypted, "Secret
     Passphrase");
document.getElementById("example1").innerHTML = encrypted;
document.getElementById("example2").innerHTML = decrypted;
```

The main purpose of the JavaScript encryption is to make the understanding of the code logic difficult even though the functionality must be unchanged. Listing 5 shows an example of using CryptoJS library in JavaScript for code encryption. Moreover, Encrypting fingerprinting scripts can help malicious web scripts to twist static analyzers and avoid being detected by anti-fingerprinting tools. For instance, Listing 6 represents a simple ”Hello World” JavaScript code before the encryption.

Listing 6. Simple Hello World JavaScript code before the encryption

```
function hi() {
  console.log("Hello World!");
}
hi();
```

Listing 7 shows the simple ”Hello World” snippet code after the encryption.

Listing 7. Encrypted Hello World JavaScript code

```
var _0x5ec0=['log'];(function(_0x83958c,_0xc60544){var
     _0x45802b=function(_0x510b72){while(--_0x510b72)
{_0x83958c['push'](_0x83958c['shift']());}};
_0x45802b(++_0xc60544);}(_0x5ec0,0x123));var
     _0x551f=function(_0x361366,_0x1e7a76)
{_0x361366=_0x361366-0x0;
var _0x31149b=_0x5ec0[_0x361366];
return _0x31149b;};function
     hi(){console[_0x551f('0x0')]('Hello\x20World!');}hi();
```

Through the runtime analyzer component, we can recognize the malicious behavior of a encrypted code; however only if some conditions are true. If those conditions are never met, the malicious behavior of the code with performing runtime analyzing cannot be spotted. A condition could be a check if the actual environment of execution is not virtualized and if this condition is false, the code will not execute. This happened with the some of the fingerprinting libraries we have analyzed (e.g., FingerprintJS, ClientJS, etc.).

Listing 8. Example code presenting how the RA module matches inline onload events via regular expressions

```
def onLoad(evestr):
regex = r"""
(onload)\s*=(\s*\"\S*\")
"""
oncl = []

matches = re.finditer(regex, evestr, re.IGNORECASE | re.MULTILINE
     | re.VERBOSE)
for matchNum, match in enumerate(matches):
matchNum = matchNum + 1

for groupNum in range(0, len(match.groups())):
groupNum = groupNum + 1
if groupNum % 2 == 0:
oncl.append(match.group(groupNum))
for val in oncl:
stri = str(val)
stri = stri.replace("\"", "")
Eventscr.append(stri)
return oncl
```

1) Analyzing encrypted codes

There are several techniques for encrypting and decoding JavaScript (JS) codes, which is explained in [38]. However, we used our innovative approach for decoding JS scripts in web pages. Our decoding techniques is optimal and does push overhead to the analysis process.

Precisely, our method is based on the monitoring of function calls inside of web pages so that the Runtime Analyzer controls whether JavaScript function calls and related access properties are related to fingerprinting.

In order to identify function calls and related to fingerprinting purposes, we have already analyzed all of the standard browser fingerprinting scripts and libraries and extracted their common used objects and methods. Whenever an encrypted JavaScript code or library, attempt to make any function call associated with any of our blueprinted functions, The Runtime Analyzer labels that encrypted code as browser fingerprinting. Moreover, because almost all standard browser fingerprinting scripts trace online users based on a unique hash code, if these functions are made, the RA identifies them as fingerprinting suspicious scripts.

Listing 9. Setting Proxy for Navigator.plugins object

```
Object.defineProperty(window, "navigator", {
value: new Proxy(window.navigator, {
get: function(target, name) {
if (name == "plugins") {
console.log("Navigator.plugins is accessed");
}
```

We perform the function call tracing via lightweight instrumenting of JavaScript Engine in our internal web browser. The method of instrumentation in FPTracker is relatively easy. It works by handling a proxy listener for all JavaScript native objects in a given website. Listing 9 shows a sample of the instrumentation for navigator.plugins object, and Listing 10 shows a sample of encrypted code which performs Plugin enumeration navigator.

Listing 10. Sample encryption function which performs Plugin enumeration

```
var _0xe2d9=["\x6C\x65\x6E\x67\x74\x68",
"\x70\x6C\x75\x67\x69\x6E\x73",
"\x50\x6C\x75\x67\x69\x6E\x73\x3A\x20",
"\x6E\x61\x6D\x65","\x3C\x62\x72\x2F\x3E",
"\x6C\x6F\x67"];
var x=navigator[_0xe2d9[1]][_0xe2d9[0]];
txt= _0xe2d9[2];
for(var i=0;i< x;i++)
{
txt+= navigator[_0xe2d9[1]][i][_0xe2d9[3]]+ _0xe2d9[4];
}
```

Lastly, overhead due to the use of cryptographic algorithms in encrypted JavaScript codes in web pages can also help us for identifying browser fingerprinting libraries. For instance, by measuring the estimated overhead based on the version of our internal web browser, we can recognize the encryption function calls and the type of browser fingerprinting libraries. Although this method is not accurate, in some cases, such as

 ISBN: 978-1-61208-729-0

large scale analysis, can speed up the investigation. Table II represents our experimental result performed by FPTracker for having the comparison of the performance between popular JavaScript libraries used for encryption in Chrome and Firefox. In this comparison, we have evaluated the performance of **SHA256** and **AES-CBC** algorithms on Chrome and Firefox web browsers. We also chose **asmcrypto**, **CryptoJS** and **SJCL** as the most widely used libraries for performing the encryption tests in JavaScript.

2) User interactions simulation

The RA module simulates user interaction on web pages via DOM objects information that has taken from the SA module (e.g., ID, Name, Object related events, etc.). The user interactions starts by scamping DOM objects in an arbitrary way. For instance, moving mouse on the web images, clicking on the form elements, entering data to the web form inputs. As an illustration in Listing 11, the RA module creates click events on three types of DOM elements to trigger JavaScript event handlers and potential hidden fingerprinting functions.

Listing 11. Three types of DOM elements

```
<button>tag
<input type = button>
<input type = submit>
```

Based on our study and experimental observation, the page load event is one of the most common places where trigger fingerprinting scripts. The scripts usually start to fingerprint visitors after some purposeful delays to avoid most of the anti-fingerprint browser extensions. These delays are made by **window.requestIdleCallback()** or **WindowOrWorkerGlobalScope.setTimeout()** JavaScript methods. The RA module also triggers document **onload** event and trace the potential fingerprinting scripts behind of this event. The RA module triggers events via **dispatchEvent** method. Listing 12 presents how the RA module triggers page onload event.

Listing 12. Sample code showing how the RA triggers the page onload event

```
var load_event = document.createEvent('Event');
load_event.initEvent('load', false, false);
window.dispatchEvent(load_event);
```

V. EXPERIMENTAL RESULTS

In this section, we summarize the results of our experiments with FPTracker. We begin by outlining our benchmark and experimental setup, describe some representative fingerprints found by our analysis and interpret the results.

In this paper, we have analyzed the top 10K European websites (according to Alexa.com). The total number of links that our tool actually reached is **1,393,426**. Our evaluation was conducted on a single Linux machine with an Intel Core i7-8500Y and 16GB memory from 7 February 2019 until end of April 2019.

A. Fingerprinting links

The total number of distinct fingerprinting URLs where FPTracker found is **117,012**, and 8% of the total reached links are browser fingerprinting while 92% are non-fingerprinting.

Moreover, **110,583** links have plain-text browser fingerprinting scripts and **6429** have encrypted scripts. In other words, **806** domains use browser fingerprinting scripts that are in plain-text, and **44** domains use the encrypted scripts (Figure 4 presents the percentages of fingerprinting links without/ with encryption).

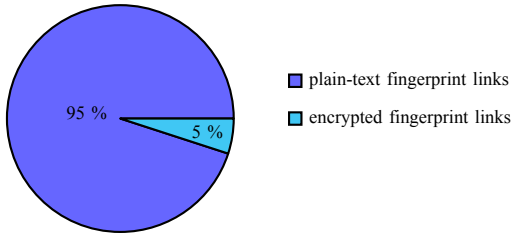

Figure. 4. Percentages of fingerprinting links without, with encryption

B. Third parties analysis

Since most of the fingerprinting scripts that FPTracker revealed are third-party libraries, it would be interesting to know about these libraries as well. Therefore, we first concentrated on identifying the most popular third parties' libraries in our benchmark. Accordingly, we determined how many distinct third-party libraries were called. In Figure 5, the 10 most used fingerprinting libraries in our evaluation are shown.

C. The adoption of the GDPR

It seems after the adoption of the General Data Protection Regulation (GDPR), websites warn users about using cookies. However, they are still interested in using browser fingerprinting scripts to collect more information about online visitors (especially in news websites which monetizing options are limited). For example, we found that **Fingerprint2** (a popular browser fingerprinting library) has been used at least in some popular crowd-sourcing websites such as **www.ebay-kleinanzeigen.de** (a popular online advertising service in German-speaking counties.).

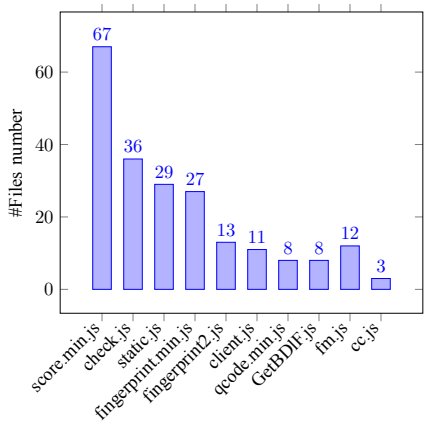

Figure. 5. The most used browser fingerprinting libraries

TABLE II. THE ANALYSIS OF THE OVERHEADS IN WEB ENCRYPTION

Cryptography Libraries	Chrome	Firefox
asmcrypto.js[1]	SHA256= 51 MiB/s AES-CBC: 47 MiB/s	SHA256: 144 MiB/s AES-CBC: 81 MiB/s
CryptoJS.js[2]	SHA256= 5.6 MiB/s AES-CBC: 3.6 MiB/s	SHA256: 28.8 MiB/s AES-CBC: 27 MiB/s
SJCL.js[3]	SHA256= 5.6 MiB/s AES-CBC: 2.35 MiB/s	SHA256: 7.2 MiB/s AES-CBC: 10.12 MiB/s

D. Fingerprinting in secondary pages

Since almost all of the previous works have studied the presence of browser fingerprinting only in the homepages of websites, the using of browser fingerprinting in the secondary pages of websites has been remained unexplored. During our experiment, we found 850 domains out of 1000 use browser fingerprinting. However, only 112 domains still use fingerprinting in the homepages, and the rest of the **688** websites fingerprint their visitors in the secondary pages. It seems that fingerprinting is becoming popular in the secondary pages of websites instead of the homepages. We presume there are two main grounds for this.

The first reason is the valuable information about users' interests (e.g. searched results) are more likely to be shown in the sub-pages (e.g., search pages, booked items, registration forms, etc.). For instance, websites such as online shopping or travel agencies can fingerprint their visitors based on their search results. The second reason can be because most of the previous large scale analysis focused only on the home pages for fingerprinting detection. Therefore, popular websites listed in Alexa.com have attempted to conceal their fingerprint scripts by placing them in the secondary pages.

TABLE III. THE MOST USED FINGERPRINT TECHNIQUES IN OUR BENCHMARK

Canvas	WebGL	WebRTC	Font Enumeration
6.7%	2.2%	0.87%	0.69%

E. Most used fingerprinting techniques

The most commonly used fingerprint techniques we found via FPTracker are Canvas, WebGL, WebRTC and Font enumeration fingerprinting (Table III). Also, the most called fingerprinting JavaScript objects in our analysis as shown in Figure 6. We have noticed that many of the domains in our benchmark, use a combination of fingerprinting techniques instead of one single method. For instance, in most domains where Canvas fingerprinting has been used, WebGL fingerprinting has existed as well.

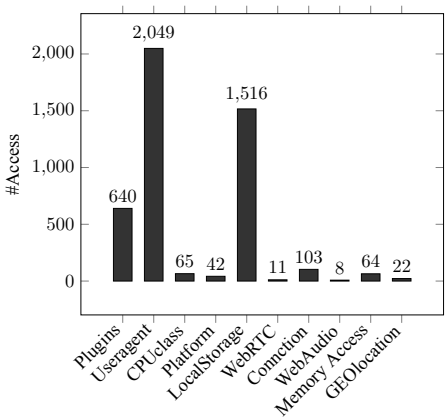

Figure. 6. The most fingerprinting access objects in the benchmark

F. Canvas fingerprinting

Our experiment recognized that **6.7%** domains out of 10,000 use Canvas fingerprinting. However, because Canvas methods can have typical applications (e.g. online gaming), we mark only those URLs as browser fingerprinting that either have any of identified fingerprinting third-party libraries or any hash computation functions related to Canvas fingerprinting. We also perceived that using Canvas fingerprinting cause about **30** milliseconds process overhead to the visitors' web browsers. Listing 13 represents an example method in FPTracker that recognizes Canvas fingerprinting. Especially, FPTracker classifies **fillText(), strokeText(), toDataURL(), getImageData()** as the most used function calls in the Canvas fingerprinting method.

Listing 13. Sample code showing how Canvas detection method in FPTracker works

```
var origToDataURL = HTMLCanvasElement.prototype.toDataURL;
HTMLCanvasElement.prototype.toDataURL = function() {
  var r = origToDataURL.apply(this, arguments);
  window.tourl++;
  return r;
  };
...
var origgetimagedata =
     CanvasRenderingContext2D.prototype.getImageData;
CanvasRenderingContext2D.prototype.getImageData = function() {
  var r = origgetimagedata.apply(this, arguments);
  window.getimagedata++;
  return r;
  };
```

G. Font detection results

Our experiment results demonstrate that **0.69%** of the domains use font enumeration fingerprinting. Enumerating installed fonts on the user web browser can have normal application for web design purposes, for instance, enumerating

fewer than 20 fonts does not indicate any web browser fingerprinting activities. However, the most of the non-fingerprinting websites enumerate fewer than 16 fonts, but some of the fingerprinting websites request between 100 to 500 fonts. The maximum number of inquired fonts in our experiment was 500 fonts (the result of Font detection is shown in Figure 7).

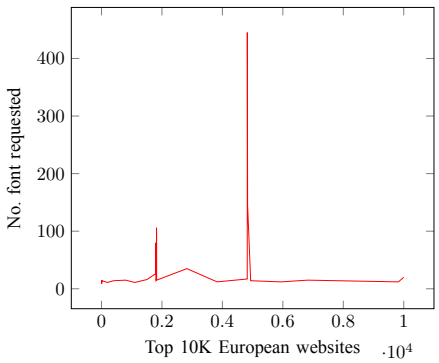

Figure. 7. Number of fonts requested for top 10K European websites. The most of the non-fingerprinting websites enumerate fewer than 16 fonts, but some of the fingerprinting websites request between 100 to 500 fonts.

We set 45 as the threshold for the number of fonts that a website can load — the threshold for the average number. Therefore, when a given website requests to load more than 45 fonts, we label its domain as sceptical about adopting font enumeration fingerprinting. FPTracker identified 16 websites that were loading more than 100 fonts. Also, during our analysis, we found that the time of font enumeration fingerprinting has more overhead in comparison with the other studied techniques introduced in this research (the average overhead for font enumeration method is approximately **90ms** on desktop Chrome).

H. WebGL fingerprinting

Based on our analysis **2.2%** of the domains use WebGL fingerprinting, using this method for fingerprinting is becoming popular since it is reliable for websites to track their users' device even if users surf the websites with several different browsers. In other words, this technique works based on user' hardware regardless of the attitudes of the web browsers. Generally, there are two methods for WebGL fingerprinting:

1) The entire of WebGL Browser Report Table will be retrieved and examined. In some cases, it is converted into a hash for faster analysis.
2) A hidden 3D image is rendered and hashed in the browser. Because the ending result depends on users' hardware which performs the computation, this method yields distinct values for different devices and drivers.

Among WebGL methods that FPTracker found in our benchmark, many of them draw a gradient with **drawArrays()** and converts it to Base64 string with toDataURL(). They also enumerate advanced WebGL extensions by **getSupportedExtensions()**. Another common extension, which interests trackers, is WEBGL_debug_renderer_info and provides information about users' graphics driver. FPTracker identifies this fingerprinting method via monitoring JavaScript function calls relate to these functions.

I. WebRTC fingerprinting

Tracing users through webRTC method is done through STUN servers. A STUN server allows users to find out their public IP address, the type of Network address translation (NAT) with a particular local port. Chrome, Firefox and Safari have implemented WebRTC which enables requests to STUN servers be made that will return the local and public IP addresses for users. In our analysis, 0.87% of the domains use WebRTC fingerprinting method and FPTracker detects the method (in Listing 14) which is commonly used for checking the compatibility of webRTC in the users' browser [10]. Moreover, we have observed that some websites use *iframe* tag to bypass webRTC blocking in client web browsers. This method is shown in Listing 15.

Listing 14. Checking the possibility to use WebRTC in Chrome, Firefox and Safari

```
var RTCPeerConnection = window.RTCPeerConnection
      || window.mozRTCPeerConnection
      || window.webkitRTCPeerConnection;
   var useWebKit = !!window.webkitRTCPeerConnection;
```

Listing 15. Using the iframe tag to bypass webRTC blocking in fingerprinting codes

```
if(!RTCPeerConnection){
  //<iframe id="iframe" sandbox="allow-same-origin"
       style="display: none"></iframe>
   //<script>...getIPs called in here...
   //
   var win = iframe.contentWindow;
   RTCPeerConnection = win.RTCPeerConnection
       || win.mozRTCPeerConnection
       || win.webkitRTCPeerConnection;
   useWebKit = !!win.webkitRTCPeerConnection;
}
```

J. Avoiding static analyzers by intentional delay

New fingerprinting libraries to avoid being detected by static analyzers do not fingerprint visitors right after page load event. Instead, they wait for a few seconds based on the visitor's web browser software (in average 30 seconds). Also, our results show that fingerprint libraries are removing font enumeration as a classic fingerprinting method due to its process overhead (between 80ms to 2000ms) that increases the chance of being detected by ad-blocker plugins (e.g., uBlock Origin, Adblock Plus, and AdBlock). The overhead of font enumeration is unusually high, especially in mobile phone web browsers (e.g. Firefox mobile edition).

TABLE IV. THE AVERAGE OVERHEAD RELATED TO COMMON FINGERPRINTING METHODS

Canvas	WebGL	WebRTC	Font Enumeration
10 ms	35 ms	30 ms	40 ms

 ISBN: 978-1-61208-729-0

VI. RELATED WORKS

Peter Eckersley is the founder of Panopticlick website. In his studies [2], three sources were used for the collection of features which are HTTP protocol, JavaScript and Flash API. Eckersley extracted 470161 fingerprints in total. There is a potential bias in collected data since these data represent those users that consider web privacy as an essential matter. Attributes such as list of installed fonts or list of installed plugins were the most dominant ones in Eckersley's data. In recent years, the capability of the JavaScript engine as an appropriate basis for identification of the client's browser type and version (even operating system) had been confirmed. Relevant benchmarks, JavaScript conformance test [6] and website rendering analysis [9], were all used for the creation of unique fingerprints. Some other techniques which are placed in the same category of fingerprinting but depend on their unique mechanism are Profiling [3], use of Evercookies [4] and History stealing [5]. Firegloves [30] is a Mozilla Firefox extension which replaces random values for the browser properties such as screen resolution. Firegloves restricts the number of available fonts on each browser tab and also returns random values for the offsetWidth and offsetHeight attributes of the HTML elements. However, it is possible to get the width and height of the HTML elements using the width and height attributes of the getBoundingClientRect method.

Another similar work is ExtensionCanvasFingerprintBlock [31] which is a browser extension for protecting users against canvas fingerprinting by returning empty images for canvas elements. Similarly, FP-Block [27] is an interesting prototype web browser extension which maintains the use and management of web identities. The web identities produced by FP-Block are distinct and logical (e.g. they are not generally randomized) that is achieved by implementing a Markov model for the generating of attribute values.

The above works diminish the fingerprinting surface by disabling browser functionalities because they do not have detection capability to perform before blocking, they block both regular and fingerprinting websites. This eventually will build annoying experience users due to the reduced functionality of the browser.

Metwalley, S. Traverso, and M. Mellia [28] proposed an algorithm based on web services that often exchange users identifiers as parameters in HTTP GET format. They look for HTTP GET requests and possible signs for the presence of user identifiers. It worth remarking that the assumption that web services transfer user identifiers through GET request is not practical because in many cases user identifiers are sent by POST queries. Another interesting work for detecting and analyzing browser fingerprinting is FPDetective [20]. They investigated the presence of fingerprinting on the home pages, using JavaScript-based font probing. FPDetective considers font-based fingerprinting as the main evidence for the existence of fingerprinting, and it proposes that fingerprinting is more popular than previously believed.

Another exciting research is [20], which analyzes the prevalence of canvas fingerprinting on the web. They discovered that Canvas fingerprinting is the most generally used tracking system which is existed in more than 5.5% of the top 100K Alexa websites. They found a total of 20 Canvas fingerprinting provider domains that are active in 5542 of the top 100K websites. FPGuard [18] is another browser fingerprinting detection tool which evaluated Alexa's top 10K focusing on Canvas and Flash-based Fingerprinting. They identified Canvas fingerprinting as the most common type of fingerprinting method on the Web. Another most recent similar works is FP-STALKER [26] which compares fingerprints to determine either they arise from the same browser instance, or they come from unknown cases. They constructed two options of FP-STALKER, a rule-based option that is fast, and a hybrid option that uses machine learning to improve precision, but it is prolonged. They identified languages and user-agent attributes as essential fingerprinting features.

Even though FPTracker similar to the previous works analyzes the presence of fingerprinting methods in websites, there are some remarkable distinctions between our work and these studies. For instance, these works have not analyzed encrypted scripts and libraries that potentially can be used for suspicious purposes (such as browser fingerprinting). Moreover, none of these previous works measures WebGL fingerprinting (which is a powerful and popular fingerprinting method these days). Moreover, during our studies we noticed that most of the state-of-the-art browser fingerprinting/ user tracking libraries use anti-crawler mechanisms to avoid Internet bots and spoofed fingerprints. The principle of these mechanisms is based on hiding their fingerprinting related functions unless online users interact with web pages. Thus, FPTracker is the first tool that traces and analyzes the professional browser fingerprinting scripts through reproducing user interactions on web pages (which help us to achieve more concrete results.). Furthermore, it seems that placing fingerprinting scripts in secondary web pages (instead of home pages) is becoming popular. This is because popular websites may attempt to evade their browser fingerprinting actions to be caught by anti browser fingerprinting systems and analysis. Websites may also desire to gather more information about their visitors based on their search results and favorite items which usually are placed on sub pages (like shopping pages). All of the works mentioned above only analyze homepages; however, FPTracker investigates all web pages including the homepage and sub pages of each website.

TABLE V. THE COMPARISON BETWEEN FPTracker AND SIMILAR WORKS

Tools	FPTracker	FP-STALKER	FPDetective	FPGaurd
JavaScript Objects	✓	✓	✓	✓
Canvas	✓	✓	-	✓
Font Enumeration	✓	✓	✓	✓
WebGL	✓	✓	-	-
WebRTC	✓	-	-	-
Encrypted Methods	✓	-	-	-
User Interaction	✓	-	-	-

 ISBN: 978-1-61208-729-0

Finally, in order to demonstrate the usefulness of our proposed approach, we compare FPTracker with other standard works. Accordingly, we have selected **FPDetective**, **FPGuard** and **FP-STALKER** as three well-known tools with similar functionality with our proposed tool. The result of the comparison (shown in Table 4) presents the advantages of our work over these tools.

VII. LIMITATIONS AND FUTURE WORKS

A. Server-side hash calculation

Even though in the past, browser fingerprinting were done mostly in the user web browser through Flash, JavaScript, HTML and CSS. However, our analysis indicates that the new browser fingerprinting libraries try to calculate the user profile hash IDs in the server side. Therefore, they use JavaScript agents that collect the value of the browser APIs and send them to server-side applications intimately.

This technique not only preserves the fingerprint libraries from being detected by anti-tracing systems, but also it protects these methods from reverse engineering and spoofing of fingerprints by Internet bots.

We have not supported this approach in this current version of FPTracker yet. However, we are working on potential solutions for tracking the potential fingerprinting data flows related to the third party servers as a future extensions of our work.

B. New applications of fingerprinting

Based on our experiments, browser fingerprinting methods are becoming popular in anti-phishing and fraud detection systems, especially for online financial and communication services.

Another advantageous application of this technique can be in detecting unusual activities on websites. For example, activities such has account harvesting, DOS attacks or identifying malicious users and Internet bots (automatically fill up web forms, or crawl the websites' information). For instance, during our studies we have noted that one of the European banks uses browser fingerprinting in one of its online services, we think that they might use it for the fraud detection purposes.

C. Client-server architecture for FPTracker

Our proposed tool is implemented as a standalone browser. Since the analyzing process can be time and memory consuming, we want to design a future extension with a client-server architecture. By doing this, a central server is responsible for patterning new fingerprint techniques, and clients use the detection functions to indicate whether a particular website comprises fingerprinting codes. Moreover, the results from clients can be used to retrain as a function for increasing accuracy. The important benefit of this plan can be detecting other type of threats on websites such as crypto-jacking, which is widespread presently.

D. Network sniffing

Another interesting extension for FPTracker would be shifting the analysis from web browser to network sniffer, to have data also on real navigation cases. Since the calls to the tracker are made from the browser, it would be essential to understand the output of the requests sent to the tracker domains, to recognize it in HTTP packets. It is worthwhile to know whether encrypted scripts can also be found in the network traffic and not only in the users' web browser.

VIII. CONCLUSION

In this research, we presented FPTracker as a standalone and practical tool for fingerprinting detection by the combination of static and runtime analysis on a large set of websites. To implement our tool, we used Selenium and Chrome driver as an instrumented standalone web browser. Moreover, our tool is able to identify encrypted fingerprints with a lightweight instrumentation approach. FPTracker also conducts user interactions on web pages to trace hidden fingerprinting scripts that used anti-crawler mechanisms. Finally, we evaluated FPTracker on the top 10K European websites including 1,393,426 links, and we have proved that standard techniques, such as font and plugin enumerations, are substituting with new technologies such as WebGL and WebRTC.

ACKNOWLEDGEMENT

The author would like to thank Prof. Dr. Christoph Kreitz for his helpful guidance and excellent comments during the review process, and the anonymous reviewers for sharing their valuable reviews.

References

[1] A. Hintz, *"Fingerprinting websites using traffic analysis."*, In Proceedings of the 2nd international conference on Privacy enhancing technologies, pp. 171-178. Springer, Berlin, Heidelberg, 2002.

[2] P. Eckersley, *"How Unique is Your Web Browser?"*, In International Symposium on Privacy Enhancing Technologies Symposium, pp. 1-18. Springer, Berlin, Heidelberg, 2010.

[3] T. Paulik, A.M. Foldes, G.G. Gulyas, *"Blogcrypt: Private Content Publishing on the Web."*, In 2010 Fourth International Conference on Emerging Security Information, Systems and Technologies, pp. 123-128. IEEE, 2010.

[4] Evercookie – virtually irrevocable persistent cookies, samy.pl/evercookie/, [retrieved: August, 2011]

[5] J. Grossman, I know where you have been, jeremiahgrossman.blogspot.com, [retrieved: August, 2011]

[6] P. Reschl, M. Mulazzani, M. Huber, and E. Weippl, *"Poster abstract: Efficient browser identification with javascript engine fingerprinting"*, In Proc. of Annual Computer Security Applications Conference (ACSAC). 2011.

[7] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham, *"Fingerprinting information in javascript implementations."*, In Proceedings of W2SP, vol. 2, no. 11. 2011.

[8] K. Boda, Á.M. Földes, G.G. Gulyás, and S. Imre, *"User Tracking on the Web via Cross-Browser Fingerprinting."*, In Nordic conference on secure it systems, pp. 31-46. Springer, Berlin, Heidelberg, 2011.

[9] K. Mowery, and H. Shacham, *"Pixel perfect: Fingerprinting canvas in HTML5."*, Proceedings of W2SP (2012): 1-12.

 ISBN: 978-1-61208-729-0

[10] A. Reiter, and A. Marsalek, *"WebRTC: your privacy is at risk."*, In Proceedings of the Symposium on Applied Computing, pp. 664-669. ACM, 2017.

[11] T. Unger et al., *"Shpf: Enhancing http (s) session security with browser fingerprinting."*, In 2013 International Conference on Availability, Reliability and Security, pp. 255-261. IEEE, 2013.

[12] V. Bernardo and D. Domingos, *"Web-based Fingerprinting Techniques."*, In Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016), V. 4, SECRYPT, pp. 271-282, 2016.

[13] P. Laperdrix, W. Rudametkin, and B. Baudry, *"Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints."*, In 2016 IEEE Symposium on Security and Privacy (SP), pp. 878-894. IEEE, 2016.

[14] S. Luangmaneerote, E. Zaluska, and L. Carr, *"Inhibiting browser fingerprinting and tracking."*, In 2017 ieee 3rd international conference on big data security on cloud (bigdatasecurity), ieee international conference on high performance and smart computing (hpsc), and ieee international conference on intelligent data and security (ids), pp. 63-68. IEEE, 2017.

[15] T. Saito et al., *"Tor Fingerprinting: Tor Browser Can Mitigate Browser Fingerprinting?"*, In International Conference on Network-Based Information Systems, pp. 504-517. Springer, Cham, 2017.

[16] P. Laperdrix, *"Browser Fingerprinting: Exploring Device Diversity to Augment Authentification and Build Client-Side Countermeasures."*, PhD diss., Rennes, INSA, 2017.

[17] N.M. Al-Fannah and W. Li, *"Not all browsers are created equal: comparing web browser fingerprintability."*, In International Workshop on Security, pp. 105-120. Springer, Cham, 2017.

[18] A. FaizKhademi, M. Zulkernine, and L. Weldemariam, *"FPGuard: Detection and prevention of browser fingerprinting."*, In IFIP Annual Conference on Data and Applications Security and Privacy, pp. 293-308. Springer, Cham, 2015.

[19] N. Takei, T. Saito, K. Takasu, and T. Yamada, *"Web browser fingerprinting using only cascading style sheets."*, In 2015 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA), pp. 57-63. IEEE, 2015.

[20] G. Acar et al., *"FPDetective: dusting the web for fingerprinters."*, In Proceedings of the 2013 ACM SIGSAC conference on Computer communications security, pp. 1129-1140. ACM, 2013.

[21] C. Neasbitt et al., *"Webcapsule: Towards a lightweight forensic engine for web browsers."*, In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 133-145. ACM, 2015.

[22] K. Singh, A. Moshchuk, H.J. Wang, and W. Lee, *"On the incoherencies in web browser access control policies."*, In 2010 IEEE Symposium on Security and Privacy, pp. 463-478. IEEE, 2010.

[23] S. Avasarala, *"Selenium WebDriver practical guide."*, Packt Publishing Ltd, 2014.

[24] G. Nakibly, G. Shelef, and S. Yudilevich, *"Hardware fingerprinting using HTML5."*, arXiv preprint arXiv:1503.01408 (2015).

[25] G. Acar et al., *"The web never forgets: Persistent tracking mechanisms in the wild."*, In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 674-689. ACM, 2014.

[26] A. Vastel, P. Laperdrix, W. Rudametkin, and R. Rouvoy, *"FP-STALKER: Tracking browser fingerprint evolutions."*, In 2018 IEEE Symposium on Security and Privacy (SP), pp. 728-741. IEEE, 2018.

[27] C.F. Torres, H. Jonker, and S. Mauw, *"FP-Block: usable web privacy by controlling browser fingerprinting."*, In European Symposium on Research in Computer Security, pp. 3-19. Springer, Cham, 2015.

[28] H. Metwalley, S. Traverso, M. Mellia, S. Miskovic, and M. Baldi, *"The online tracking horde: a view from passive measurements."*, In International Workshop on Traffic Monitoring and Analysis, pp. 111-125. Springer, Cham, 2015.

[29] S. Englehardt, and A. Narayanan, "Online tracking: A 1-million-site measurement and analysis.", In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pp. 1388-1401. ACM, 2016.

[30] K. Boda, fingerprint.pet-portal.eu/?menu=6, [retrieved: March, 2019]

[31] Appodrome, Canvasfingerprintblock, http://goo.gl/1ltNs4, [retrieved: December, 2018]

[32] D. Fifield, and M.G. Epner, *"Fingerprintability of WebRTC."* arXiv preprint arXiv:1605.08805 (2016).

[33] chromedriver, sites.google.com/a/chromium.org/chromedriver, [retrieved: December, 2018]

[34] statcounter, gs.statcounter.com/browser-market-share, [retrieved: December, 2018]

[35] beautifulsoup4, pypi.org/project/beautifulsoup4, [retrieved: February, 2019]

[36] panopticlick, panopticlick.eff.org, [retrieved: January, 2019]

[37] M. Juarez et al., "A critical evaluation of website fingerprinting attacks.", In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 263-274. ACM, 2014.

[38] M. AbdelKhalek, and A. Shosha, "Jsdes: An automated de-obfuscation system for malicious javascript.", In proceedings of the 12th International Conference on Availability, Reliability and Security, p. 80. ACM, 2017.

 ISBN: 978-1-61208-729-0

GPS Spoofing for Android and iOS Mobile Systems

Michael Nelson
Computer Science, Iona College
New Rochelle, USA
email: mnelson4@gaels.iona.edu

Paolina Centonze
Computer Science, Iona College
New Rochelle, USA
e-mail: pcentonze@iona.edu

***Abstract*— Global Positioning Systems (GPS) are incredibly commonplace in daily life. Examples include fishing vessels, aircraft, cars, and even mobile phones. Currently, civilian GPS signals are transmitted in plain text on known frequencies. Due to the lack of security and obfuscation, there have been many experiments proving the ease and dangers of GPS spoofing and jamming. However, most of the analysis in the past few years has focused on vehicles rather than the devices everybody has and uses daily: smartphones and tablet devices. Spoofing these devices through satellite defined radio as well as applications will show the biggest vulnerabilities in these systems. As of now, there is no clear evidence of one Operating System (OS) being more secure than another or what types of applications are most susceptible to hacking. In this paper, we analyze the effectiveness of different spoofing methods on various commonly used applications. This analysis will determine whether iOS or Android is easier to spoof by comparing the ease and effectiveness of these spoofing methods.**

Keywords- GPS; spoof; mobile; iOS; Android.

I. INTRODUCTION

Several famous cases of GPS spoofing and jamming have come out in the past decade. Vehicles such as military drones and mega yachts [1] alike have been hacked into going to an unintended location using vulnerabilities in their navigation systems. With an increase of devices using GPS and location-based services, the severity of false locations will only become more widespread. While it is easy to see why a person would spoof another, whether it be to draw them to a remote location, land their equipment on another's property, etc., what many people overlook is why someone would spoof their own signal. Fishing vessels do it to fish in restricted waters, drone users do it to fly in unauthorized air space, and media consumers do it to access region-blocked content. Whatever the reason for spoofing, the intent is malicious. Right now, GPS messages for the public are called Civilian Navigation (CNAV) messages [2]. These civilian navigation messages have a warning that they should not be used for safety-of-life or other critical purposes until they are declared safe by the government. Even the government sees clear flaws and vulnerabilities in the current system. They are producing new types of messages regularly to improve GPS systems, however, for something that is used so much in many devices, it should be a top priority to make secure and robust. Many sectors of the economy and life rely on GPS [3]. A forced landing of a US drone by Iran is discussed in detail in "Susceptibility of GPS-Dependent Complex Systems to Spoofing" [3] that was performed despite the encryption and protection of military GPS. Others have tested spoofing in different types of scenarios, such as with car GPS devices in "A Practical GPS Location Spoofing Attack in Road Navigation Scenario" [6]. This analysis has a different set of challenges to overcome. Unlike a boat, that has a wide open sea, cars will notice if their GPS directs them to drive off the road. For this setup, a car has to actively follow another car and make small changes to the route as it travels. While unique, it is just one of many examples of practical GPS spoofing that can affect an average person. Unlike the analyses of the papers just mentioned, our work goes more in-depth into mobile OS spoofing. Multiple spoofing methods, as well as applications, will be compared to prove where the biggest security flaws are.

The rest of the paper is structured as follows. Section 2 will go over the technical aspects of how location and time are calculated from GPS signals as well as the process of spoofing a GPS receiver. Section 3 will discuss similar works that act as a proof of concept for mobile GPS spoofing. Next, Section 4 will go over how the experiment and analysis will be performed. It will also include details on the software and hardware that will be used. Section 5 will present some preliminary results from what has been done so far. Finally, Section 6 will list some conclusions that can be drawn at this stage and how future experiments can build off of this work.

II. TECHNICAL BACKGROUND

The basic concept of how GPS spoofing works is simple. Normally, a receiver is connected to several satellites, as shown in Figure 1. The spoofer sends out a weak signal that becomes more and more powerful until it overtakes the signals from the satellites. Once locked to the receiver, false signals can be sent to the device to make it think it needs to change course to correct itself. The light blue line in Figure 2 shows the course the device or vehicle was supposed to take, and the solid line shows where it actually ends up. Whether it be a drone, boat, car, or phone, this concept is the same.

A basic understanding of how the GPS signals are sent and received is also needed to understand how spoofing works. Figure 1 does not have a random number of satellites. At any given time, a receiver is using signals from four different satellites, as shown. Each of these signals is processed with a delay for the time it takes for a signal to get from the satellite to a receiver. The delay is calculated with the individual signals to coordinate time and position. The calculations are as follows:

 ISBN: 978-1-61208-729-0

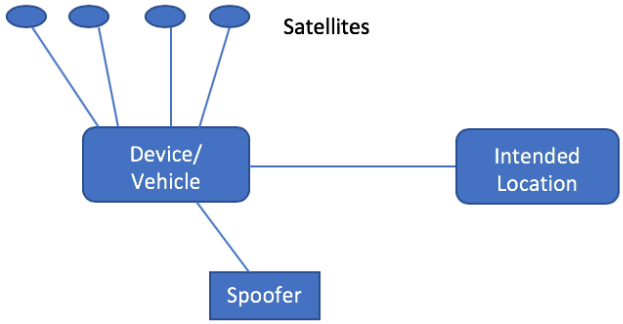

Figure 1. Phase 1.

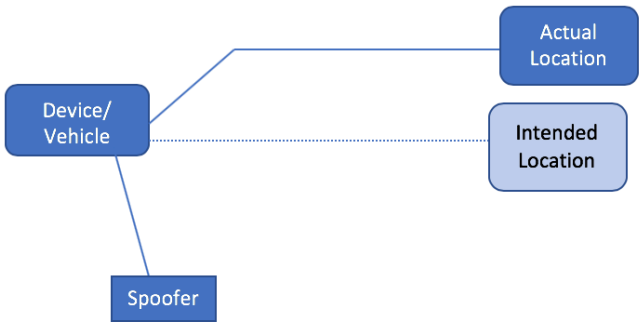

Figure 2. Phase 2.

$$Delay_i * C = Path_i \quad (1)$$

$$(T_i - T_0) * C = Position(sat_i) - position(RX) \quad (2)$$

$$(T + D_1 - T_0) * C = Pos(x_1, y_1, z_1) - Pos(x, y, z) \quad (3)$$

$$(T + D_2 - T_0) * C = Pos(x_2, y_2, z_2) - Pos(x, y, z) \quad (4)$$

$$(T + D_3 - T_0) * C = Pos(x_3, y_3, z_3) - Pos(x, y, z) \quad (5)$$

$$(T + D_4 - T_0) * C = Pos(x_4, y_4, z_4) - Pos(x, y, z) \quad (6)$$

The x, y and z variables with number subscripts represent the coordinates of the satellites, while the x, y and z without subscripts are the receiver coordinates. Modifying these signals with the expected delays is what allows a GPS device to be spoofed. The delays are all the D numbered variables in the equations. T0 is the initial time of the request while T is the current time. All that is needed to know about C is that it is a constant. As pointed out by [4], the proper delays are the most difficult part of GPS spoofing. Producing GPS signals is not that hard, but getting them to transmit with the proper delays to represent both the movement of the satellites and rotation of the earth is the key to a successful spoof. The four equations (3)-(6) are combined to get an accurate reading of the receivers location and time. All equations are taken from [4].

III. RELATED WORK

Some groups have tried and succeeded in spoofing cell phones and other mobile devices after overcoming the signal delay problem. [4] failed at this at first citing the problem of the Doppler effect. Once this obstacle is overcome and the mobile device is receiving the spoofed GPS signals is where this analysis starts to differ. Instead of testing Satellite Defined Radio (SDR) spoofing on several devices, it will be tested on two devices of similar specifications alongside a different type of location spoofing. These two different spoofing methods will be tested on a variety of applications including Snapchat, Tinder, Poke´mon Go, and more. While there is published research about spoofing the location of a user in Poke´mon Go [5], the paper does not delve into how these methods work on other location-based applications. There still remains the question of whether other applications are already able to detect this spoofing and if so what makes them different. Also, the methods they try do not include SDR spoofing. The authors fail to mention whether this was out of the scope of their analysis or seemed too difficult to implement, but it is a valid method of location spoofing all the same.

IV. METHODOLOGY

Many mobile applications use the location of their users for some purpose. The objective of this analysis is to see where these location-based applications are most vulnerable and on what platform. The two spoofing methods used will be SDR, where a signal will be transmitted to the tablet, and spoofing applications that alter the location on the tablet. Most researchers until now have focused on falsifying signals and transmitting them to phones, small drones, GPS receivers, etc. So, while there is a proof of concept for spoofing mobile devices the results of this analysis should provide a deeper understanding of how they affect the applications that people use and could disrupt their daily lives. The process by which the analysis will go is as follows:

1. First, the spoofing device will be set up, ready to transmit a signal and the spoofing applications chosen
2. It will then be confirmed the devices are receiving the false GPS signal from the spoofing device
3. The testing applications will be set up and used while the spoofing is taking place
4. All results will be recorded and analyzed

A. Devices

Testing will be done on two tablets using false GPS data. One is a Nvidia Shield K1 while the other is a 32GB iPad with Wi-Fi. Having two unique and distinct operating systems, iOS and Android, will allow for comparisons on which one, in general, is easier to spoof through applications and/or SDR or if they are similar. It is possible the architecture of one OS has a key vulnerability, in the way they receive and calculate GPS signals which the results should determine. The device generating the signal for the spoof is a HackRF, one capable of operating frequencies from 1 to 600 MHz. It is USB powered, plugging directly into a computer. As seen in Figure 3, the left side is used to attach an antenna for better reception and transmission while the right side has connector options for

 ISBN: 978-1-61208-729-0

Figure 3. HackRF One.

clock input and output. It is also capable of receiving data and could be used to record and replay data it receives to another device. We use predetermined coordinates instead of receiving and transmitting them live or through a replay.

B. Software

A simulator called "gps-sdr-sim" is available on GitHub [7]. There are several simulators available online, some free and others paid. However, the cost of paid simulators is very expensive and out of the scope for the average person. One of the arguments we are trying to make is anyone can hack GPS signals, so the technology would need to be easily available to all. The software chosen is compatible with the HackRF and has several options available. One is pulling data from a file that can be prepared before the run. If other SDR devices were used, a live signal could be recorded and rebroadcast at a different time and place. Another way is to just run it with a set of coordinates and a time and it broadcasts a steady signal, as shown in Figure 4. Only the coordinates at end of the terminal command are important to us as they are the coordinates of the intended fake location.

This is the simplest way but the most effective for our intentions. The tablets only need to spoof a steady location rather than actively move around, as pointed out in [6]. An active spoof is particularly difficult because, if at any point the spoofed signal is interrupted, the GPS device will try to reconnect and possibly connect back to satellites providing real data. Each device also uses a different spoofing application as there are different online marketplaces for each OS. Android will use an application called Floater. This application is free on the Google Play Store and fairly simple to set up. On iOS, an application called iSpoofer will be tested. It costs about thirteen dollars for a three-month subscription and requires a windows computer. However, there is a free trial available which will be used to not give an unfair advantage to one application. While not as convenient as the Android application, it is still accessible and usable by the average person.

C. Performing the Spoof

All the necessary applications will be installed on both devices with either the same or as close to the same version as possible. Otherwise, while unlikely, it is possible that security

```
Michaels-MBP-4:gps-sdr-sim-master michaelnelson$ ./gps-sdr-sim -e brdc3540.14n -l 30.286502,120.032669,100
Using static location mode.
Start time = 2014/12/20,00:00:00 (1823:518400)
Duration = 300.0 [sec]
01   66.2   3.2  25487375.4  22.8
02  272.4  23.3  23740075.2  11.2
03   41.9  22.9  23445592.2  12.9
06  308.3  54.9  21093772.1   7.0
09  124.5  21.5  23554774.3  14.6
10  201.3  48.8  21181674.0   8.0
12  323.3   9.3  24728586.3  12.9
17   49.3  68.8  20618819.7   6.6
20   45.8  32.9  22428634.1  10.5
```

Figure 4. GPS Simulator Run.

holes we are exploring could have been patched in that time. Also to ensure fairness, the spoofing will be done at the same location with the same fake location. Otherwise, it would be possible that a certain area has stronger GPS signals, compromising the results.

V. EXPECTED RESULTS

Once the devices are being spoofed, several applications will be tested from categories including: streaming, social media, gaming, and possibly more. Their version number and whether they could be spoofed will all be recorded for analysis four times, once for each OS with both spoofing methods. Depending on the application, there will be different indications of a successful spoof. For example, some media streaming applications will show another countries library because of the location, but will not stream the content because it can tell that it is not the actual region of the device through other information like IP or DNS. This would mean the spoof was unsuccessful. Four tables similar to Tables 1 and 2 will be created for Android and iOS, respectively. Two tables will be for the application spoofing, and two tables will be for the SDR spoofing. These include some preliminary results for the application spoofing.

As expected, the SDR spoofing is a bigger challenge than the application spoofing due to improper delays. This is most likely due to a bad timing crystal, which is common in a simple HackRF device. Having a separate device that could offer a clock input for the HackRF could fix this. As for the results, Tables 1 and 2 have preliminary results from the application spoof. The application version numbers were recorded, which sometimes differ greatly because of the different release of updates on either OS. The "yes" or "no" in the app spoof rows signify whether the spoof was or was not successful. The applications tested can be further broken up into three categories: streaming, social media, or gaming. Poke´mon Go, Geocaching, and Turf Wars are gaming applications. Netflix and YouTube are for streaming. Snapchat and Tinder would be considered social media applications. By analyzing the results of the different categories, it is possible to see more vulnerabilities in one because of the method they extract their location data or rely on GPS. Conclusions could be drawn from the results received so far, however it is better to not make too many assumptions on incomplete data. As of now, it seems that streaming applications are less susceptible to spoofing, probably due to gathering locations from alternate methods than GPS. However, many applications were able to be spoofed on both operating systems, which displays big vulnerabilities in GPS

 ISBN: 978-1-61208-729-0

TABLE I. ANDROID SPOOF.

Application	Pokémon Go	Geocaching	Turf Wars	Netflix	Youtube	Snapchat	Tinder
Version	0.138.3	7.10.1	1.47	7.6.0	14.15.53	10.55.0.0	10.13.0
App Spoof	yes	yes	yes	no	no	yes	yes

TABLE II. IOS SPOOF.

Application	Pokémon Go	Geocaching	Turf Wars	Netflix	Youtube	Snapchat	Tinder
Version	1.109.0	7.10.0	2.981	11.31.2	14.16	10.56.0.23	10.12.1
App Spoof	yes	yes	yes	no	no	yes	yes

or in both iOS and Android operating systems. Final results should be able to give a more accurate and detailed analysis than possible right now.

VI. CONCLUSIONS AND FUTURE WORK

This analysis has shown the vulnerabilities that currently exist in mobile OSs and possibly GPS receivers. Further research could be done into why some applications can detect or block spoofing and others cannot. Once specific flaws are found, solutions and patches can be developed to ensure the security of location-dependent applications. As the users location is used in more and more systems, accuracy and security are paramount. Further analysis could be done with more spoofing applications, other SDR devices, or a different set of user applications. Multiple devices would be able to receive and transmit information in real-time, which may have a better effect than the static signals we used. There are a variety of methods and strategies that could be implemented to achieve the same result. What is important, is with the rise in the use of GPS in so many facets of life, advancement in security needs to follow or match it.

ACKNOWLEDGMENT

We are thankful for the computer science department at Iona College based in New Rochelle, New York for providing the tablets for testing as well as structuring their curriculum to encourage research.

REFERENCES

[1] M. L. Psiaki and T. E. Humphreys. “Protecting GPS From Spoofers Is Critical to the Future of Navigation.” IEEE Spectrum: Technology, Engineering, and Science News, IEEE Spectrum, 29 July, 2016.

[2] Official U.S. government information about the Global Positioning System (GPS) and related topics. [Online]. [retrieved: April, 2019] Available From: https://www.gps.gov/.

[3] L. Faria, C. A. Silvestre, M. A. Correia, and N. A. Roso. “Susceptibility of GPS-Dependent Complex Systems to Spoofing”. Journal of Aerospace Technology and Management, 10, e0218. Epub January 15, 2018 [retrieved: June, 2019] doi:10.5028/jatm. v10.839

[4] L. Huang and Q. Yang. “Low-cost GPS simulator – GPS spoofing by SDR”. Qihoo 360 Technology Co. Ltd. Defcon 23, Las Vegas. [PowerPoint slides] 2015

[5] B. Zhao and Q. Chen. “Location Spoofing in a Location-Based Game: A Case Study of Poke mon Go”. Advances in Cartography and GIScience (ICACI ’17), Lecture Notes in Geoinformation and Cartography. Springer, Cham. pp. 21-32. [retrieved: May, 2019] doi:10.1007/978-3-319-57336-6 2

[6] K. Zeng, Y. Shu, S. Liu, Y. Dou, and Y. Yang. “A Practical GPS Location Spoofing Attack in Road Navigation Scenario”. Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications. (HotMobile ’17). Feb. 2017. pp. 85-90. [retrieved: May, 2019] doi:10.1145/3032970.3032983

[7] Simulator Link. [retrieved: May, 2019] https://github.com/osqzss/gps- sdr- sim

[8] Device Site Link. [retrieved: June, 2019] https://greatscottgadgets.com/hackrf/one/

[9] YouTube Application Link. [retrieved: June, 2019] https://www.youtube.com/

[10] Poke´mon Go [retrieved: May, 2019] Application Link. https://www.pokemongo.com/en-us/

[11] Netflix Application Link. [retrieved: June, 2019] https://www.netflix.com/browse

[12] Tinder Application Link. [retrieved: June, 2019] https://tinder.com/app/recs

[13] Snapchat Application Link. [retrieved: June, 2019] https://www.snapchat.com/

[14] Geocaching Application Link. [retrieved: June, 2019] https://www.geocaching.com/play/mobile

[15] Turf Wars Application Link. [retrieved: June, 2019] https://turfwarsapp.com/

 ISBN: 978-1-61208-729-0