ICIMP 2013

The Eighth International Conference on Internet Monitoring and Protection

ISBN: 978-1-61208-281-3

June 23 - 28, 2013

Rome, Italy

ICIMP 2013 Editors

William Dougherty, Secern Consulting - Charlotte, USA

Petre Dini, Concordia University, Canada / China Space Agency Center, China

ICIMP 2013

Forward

The Eighth International Conference on Internet Monitoring and Protection (ICIMP 2013) held on June 23 - 28, 2013 - Rome, Italy, 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.

Internet and Web-based technologies led to new frameworks, languages, mechanisms and protocols for Web applications design and development. Interaction between web-based applications and classical applications requires special interfaces and exposes various performance parameters.

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.

We take this opportunity to thank all the members of the ICIMP 2013 Technical Program Committee as well as the numerous reviewers. The creation of such high-quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to the ICIMP 2013. We truly believe that, thanks to all these efforts, the final conference program consists of top quality contributions.

This event could also not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the ICIMP 2013 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that ICIMP 2013 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in Internet Monitoring and Protection

We are convinced that the participants found the event useful and communications very open. We also hope the attendees enjoyed the historic charm Rome, Italy.

ICIMP 2013 Chairs

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

ICIMP 2013

Committee

Table of Contents

Assessment of Cybersecurity Knowledge and Behavior: An Anti-phishing Scenario

Ping An Wang
Department of Cybersecurity and Information Assurance, Graduate School
University of Maryland University College
Adelphi, Maryland, USA
Email: pwang2050@yahoo.com

***Abstract* – This paper focuses on the cybersecurity risk of online phishing and anti-phishing solutions to study user knowledge of phishing risks and intention to use anti-phishing solutions. Online phishing has become a major avenue for cyberattacks and a common cause for identity theft and financial losses. The availability of online security and anti-phishing solutions has been on the rise. However, there has been little research focus and consensus on assessing the effect of users' knowledge of cybersecurity risks and technology solutions on their acceptance and adoptions of cybersecurity solutions. This study proposes a novel model of assessment of users' cybersecurity knowledge and acceptance, which consists of both direct objective assessment and indirect self-assessment. The research model is designed to evaluate the relationship between online users' technical knowledge and competence in cybersecurity risks (i.e., online phishing risks) and their behavioral intention to use cybersecurity (i.e., anti-phishing) technology solutions. This study employs a survey method that measures end users' knowledge and competence of anti-phishing techniques and their intention to adopt and use anti-phishing solutions. Statistical analysis of the data collected suggests a positive correlation between users' technical knowledge of online phishing risks and solutions and their intention to adopt and use anti-phishing solutions. A positive correlation also appears between the direct assessment answers and self-assessment responses.**

Keywords-cybersecurity; phishing; knowledge; assessment

I. INTRODUCTION

Cybersecurity risks such as online phishing, have become an increasingly significant issue for information technology (IT) end users. Online phishing is a common cybersecurity attack targeting unsuspecting users and victims who are lured into clicking a spoofed universal resource locator (URL) or fraudulent email attachments or links pointing to a rogue Web page to give away their sensitive personal and financial information. The latest Phishing Activity Trends Report released by Anti-phishing Working Group (APWG) shows that attacks targeting consumers have remained at high levels with hundreds of phishing websites established online every day to lure online users to trouble and loss [1]. A recent report from security firm Trend Micro also confirms that 91 percent of cyberattacks now start with spear phishing, a special type of phishing targeting specific individuals [2].

There have been considerable efforts from cybersecurity industry and experts to make countermeasures and technology solutions available to detect, prevent, and minimize losses from cybersecurity attacks. Anti-phishing technology solutions do exist, which include anti-phishing features built in web browsers and protection against phishing in commercial software from cybersecurity vendors, such as Symantec. However, the latest CSI Computer Crime and Security Survey report indicates a consistently small investment and effort in end user cybersecurity awareness training for several years [3]. Meanwhile, users' adoption and actual use of cybersecurity solutions have been relatively low and not commensurate with the severity level of their perceptions and concerns about cybersecurity risks [4, 5]. Thus, it is vital to assess end users' knowledge of cybersecurity risks and the key factors in users' decision on adopting and using cybersecurity solutions.

Human capital and cybersecurity knowledge are the essential factors for achieving technical competence in the general cybersecurity competency model [6]. Knowledge is the contextual and high-value form of information and experience ready to apply to decisions and actions [7]. Research findings have indicated IT users' lack of knowledge and clear understanding of cybersecurity solutions, including protections against phishing, unwanted tracking of their online activities, and potential leak of sensitive personal information [8, 9, 10]. However, there has been little research on user acceptance of cybersecurity solutions from the knowledge perspective. Most of the prior studies on technology acceptance focused on new technologies in general and the constructs of user perceptions of usefulness and ease of use as determinants of user attitude and behavioral intention toward technology adoption. These studies were primarily based on the technology acceptance model (TAM) and the theory of planned behavior (TPB). The research model proposed in this study focuses on the relationship between the assessment of user knowledge of cybersecurity and user attitude and intention toward adopting and using cybersecurity solutions.

Empirically, this study uses a behavioral survey method to assess users' knowledge and attitude and intention regarding cybersecurity (i.e., anti-phishing) technology solutions. The findings suggest that user knowledge is positively associated with user acceptance of and intention to use in the domain of cybersecurity (i.e., anti-phishing) technology.

There are six sections in this paper. Section I introduces the paper and the motivation for the study. Section II explains the research goals and discusses relevant theoretical background. Section III presents the research model. Section IV explains the survey-based methodology used for this study. Section V presents the results and data analysis. Section VI concludes the paper with findings, implications, and possible follow-up research.

II. RESEARCH GOALS AND BACKGROUND

The goal of this research is to use the anti-phishing scenario to assess end user knowledge of cybersecurity risks and solutions and uncover the relationship between user knowledge and user acceptance of cybersecurity solutions. Knowledge includes contextual information, awareness, and personal experience ready to be used for decisions and actions. Knowledge consists of both explicit knowledge or communicable information and tacit knowledge, which is personal and intuitive insights and know-how originated from individual experiences and values [11]. One's attitude is a determining factor of one's behavioral intention that predicts one's actual behavior [12]. User knowledge of online security risks and protective and preventive solutions was found to be an important factor affecting user attitude and trust in online vendors in the e-commerce domain [13, 14]. Thus, users' knowledge of cybersecurity may be a predictor of their attitude and intention that are essential to their acceptance of security solutions. This research study primarily explores the relationship between user knowledge of cybersecurity and user acceptance of cybersecurity solutions. In addition, this study attempts to address the question of how to properly assess user knowledge of cybersecurity risks and technology.

A. Attitudinal Theories on Cybersecurity Knowledge

A large amount of prior research on cybersecurity knowledge was based on attitudinal theories involving users' perceptions of online risks and behavioral intentions. The conceptual assumption of such models was based on the theory of reasoned action (TRA). In TRA, behavioral intentions are antecedents to individual behavior, and intention is determined by attitudes and perceptions [12]. Accordingly, one's perception and attitudes regarding online risks will have an influence on attitudes toward online transactions and in turn, affect his or her behavioral intentions to conduct online transactions.

Several studies used attitudinal theories to study how risk perceptions affect a dependent variable such as trust, purchase intention, and etc. These studies include Bhatnagar et al. [15], Miyazaki and Fernandez [16], Salisbury et al. [17], Pavlou [18], Milne et al. [19], Dinev and Hu [20], Jiang et al. [14], and Tsai et al. [21]. Knowledge in these studies generally refers to experience, maturity of subject, user awareness in a general sense, and familiarity with a task or risks in the online purchase environment.

The primary interest of these attitudinal studies was in user perceptions of online risks and intention to purchase online. Their common assumption is that people's decisions under risks are driven by inconsistent perceptions, beliefs, and emotions. Such an assumption does lend support for the suggestion that users' self-assessment of knowledge may have an impact on their behavior and decision making. However, the attitudinal studies have two major limitations: a) no focus on the assessment of user knowledge of cybersecurity risks and solutions; b) no focus on the relationship between cybersecurity knowledge assessment and intention to adopt cybersecurity technology solutions. This research attempts to address these limitations.

B. Psychometric Theories on Cybersecurity Knowledge

The psychometric paradigm is an important approach to the knowledge dimension of risk studies even though such research on online security risks has been limited. The psychometric paradigm uses multivariate techniques to recover cognitive maps of decision makers' risk perceptions and attitudes so as to understand the dimensions of risk and the cognitive schema [22]. Fischhoff et al. studied technological risks and benefits using the psychometric paradigm with some inclusion of knowledge of risks, but the study did not address the relationship between risks and technology acceptance [23]. Slovic, Fischhoff, and Lichtenstein found that risk acceptability is affected by risk attributes, such as familiarity with the level of risk [22]. They equated the concept of knowledge of risk to people's familiarity with the level of risk. However, the study did not address the assessment of cybersecurity knowledge or user acceptance of online security technology solutions.

Slovic further elaborated the psychometric approach to the study of risk perceptions and suggested that the level of knowledge attribute seems to influence the relationship between perceived risk, perceived benefit, and risk acceptance [24]. However, he did not clearly define the concept of knowledge and did not include cybersecurity knowledge and technology acceptance in the study.

Nyshadham and Ugbaja used the techniques of the psychometric paradigm to study how B2C e-commerce consumers organize novel online risks in memory. The study called for further analysis to define the risk dimensions [25]. Using the psychometric paradigm, Gabriel and Nyshadham studied perceptions of online risks that affect online purchase intentions [26]. The study contributed a valuable taxonomy of online risks and a cognitive map of online consumers' risk perceptions and attitudes. This is a positive step toward recognizing the knowledge dimension in user perceptions of online risks in general. However, there was no focus on cybersecurity risks and technology acceptance in the study.

C. Knowledge and Technology Acceptance

Prior literature on user knowledge and acceptance of cybersecurity solutions was primarily based on TPB and TAM. TPB considers user attitude, perceived social norm, and perceived behavioral control to be the factors determining user behavioral intention that predicts actual user behavior of adopting technology solutions [27]. TPB was the theoretical basis for the anti-spyware adoption model [4]. However, the model did not address the user knowledge factor. Another study on user attitude toward spyware and anti-spyware technologies was based on TPB and TAM [28]. In addition to the three TPB constructs, the TAM constructs of perceived usefulness and perceived ease of use were included as predictors of user attitude and behavior toward anti-spyware solutions. The study did include computer knowledge and awareness of spyware as predictors of user action. However, neither the construct of knowledge nor its role in the TPB model was clearly defined or assessed. Also, these two studies on spyware focused on anti-spyware technology, without addressing phishing risks and anti-phishing solutions.

A later study on protective information technologies based on an extended model of TPB attempted to address user behavioral intention toward cybersecurity technology in general [20]. The extended TPB model dropped the perceived usefulness and perceived ease of use factors and emphasized user awareness as a key determinant of user intention toward adopting protective information technologies. However, as acknowledged by the study itself, the awareness construct was not specific. The survey approach used by the study was also limited to questions on spyware and anti-spyware solutions.

A different theoretical approach to user acceptance of cybersecurity solutions was based on the Protection Motivation Theory (PMT) [29]. The PMT model argues that one's fear appeals affect the motivation to protect oneself from potential harm. The study concluded that perceived vulnerability, perceived severity, response efficacy, and response cost influence individual behavioral intention to adopt anti-spyware protective technology. However, the PMT model did not address the user knowledge factor or assessment of user knowledge of cybersecurity. Subsequent studies by Wang [30] and Wang and Nyshadham [31] contributed more in-depth definitions of the user knowledge constructs regarding online risks, but their studies focused on the effect on online purchase intentions and decisions with little emphasis on intentions to adopt online security technology solutions.

III. RESEARCH MODEL

Based on the review of the existing research above, this study proposes a new model of technology acceptance to address user acceptance of cybersecurity solutions from the knowledge assessment perspective. This model, shown in Fig. 1, extends the traditional TAM theory to include and focus on the knowledge assessment factor in determining user attitude and intention in the specific anti-phishing cybersecurity context. Knowledge management theory defines knowledge as the contextual and high-value form of information and experience that positively affect decisions and actions [32]. User knowledge in this study includes explicit contextual information and awareness as well as tacit personal experience and technical know-how. In terms of basic knowledge of anti-phishing solutions, users should be aware of and look for the https secure protocol in the URL for a secure website as shown in a sample secure URL in the browser window in Fig. 2. As an indicator of more in-depth knowledge of anti-phishing technology, users should look for a valid security certificate for a secure website as shown in Fig. 3.

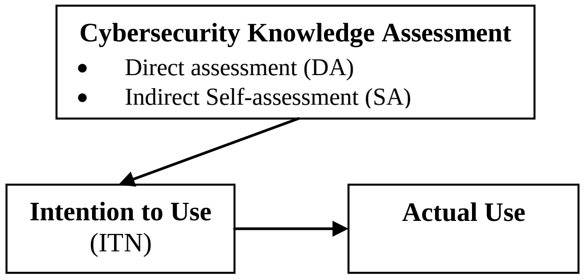

Cybersecurity Knowledge Assessment
• Direct assessment (DA)
• Indirect Self-assessment (SA)
Intention to Use (ITN)
Actual Use

Figure 1. Research Model

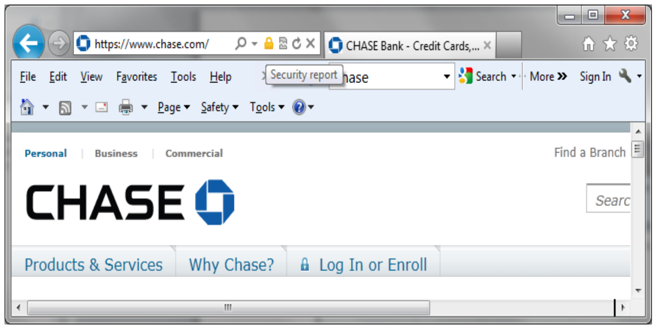
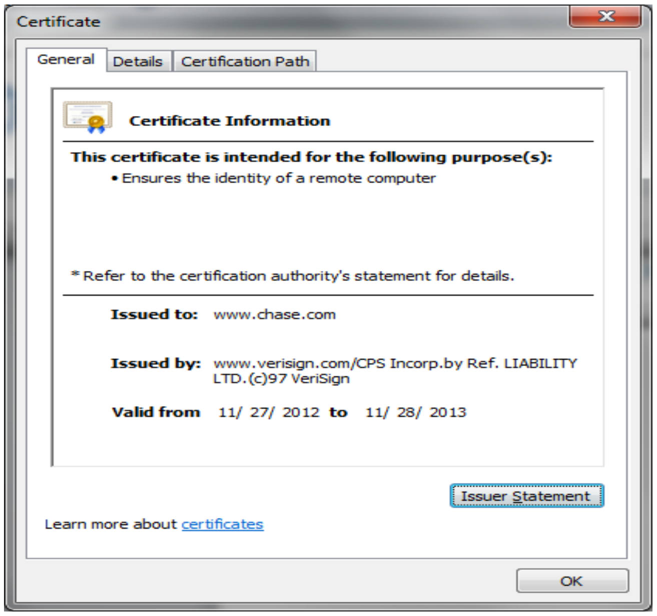


Figure 2. Secure non-phishing URL with https protocol

Assessment of knowledge is a process of measuring and comparing learning expectations and actual learning outcomes [33]. Knowledge assessment in information technology areas can include both direct assessment, such as objective tests and projects, and indirect assessment, such as perceptions and self-assessment (or self-evaluation), and in evaluating the outcomes of technical knowledge and skills both direct and self-assessment methods can work together with positive correlation in results [34]. Accordingly, in assessing user knowledge of phishing risks and anti-phishing technology solutions, objective test questions on technical indicators for secure websites can be used as direct assessment, along with indirect self-assessment questions for users to present their perceptions and self-evaluations of their anti-phishing knowledge and skills.


Certificate

General | Details | Certification Path

Certificate Information

This certificate is intended for the following purpose(s):
• Ensures the identity of a remote computer

* Refer to the certification authority's statement for details.

Issued to: www.chase.com

Issued by: www.verisign.com/CPS Incorp.by Ref. LIABILITY LTD.(c)97 VeriSign

Valid from 11/ 27/ 2012 to 11/ 28/ 2013

Issuer Statement

Learn more about certificates

OK

Figure 3. Valid certificate for a non-phishing website

User acceptance refers to user's positive attitude and intention toward using the cybersecurity solutions they know. One's attitude, according to TRA and TAM, determines one's behavioral intention which in turn predicts one's actual behavior. Therefore, the research model proposes that user knowledge of cybersecurity risks and solutions is positively related to user attitude and intention toward using cybersecurity solutions.

TAM has been a traditional model for studying computer usage behavior and acceptance of information technology in general. However, to focus on the knowledge variable, the research model in this study does not include the TAM constructs of perceived usefulness (PU) and perceived ease of use (POEU). The awareness (a component of knowledge) of the consequences of not using cybersecurity technologies is more significant than PU and POEU in affecting user attitude and intention [20]. The constructs of intention to use and actual use are defined according to TAM constructs [35]. Intention to use measures the strength of one's decision to perform the action or behavior of using the cybersecurity technology. Actual use refers to the actual action or behavior of using the cybersecurity technology.

IV. METHODOLOGY

To test the research model, an anonymous questionnaire-based survey was conducted among 210 randomly selected undergraduate students of various majors at a college in the northeast of the United States. The assumption for such randomization is that students of certain technical majors may have more cybersecurity background than those of non-technical majors. Self-report survey question format has been an effective method for eliciting attitudinal responses [36]. The survey for this study utilized a seven-point Likert scale of responses ranging from 7=Strongly Agree to 1=Strongly Disagree.

There are eight content questions related to user knowledge of phishing risks and anti-phishing technology solutions. The content of the questions is based on some common phishing risks and solutions for online activities and transactions. The first four questions in the survey are direct assessment questions expecting factual answers from the subjects to measure their knowledge of phishing risks and essential anti-phishing technology. Questions 5 through 7 are indirect self-assessment questions to elicit the respondents' self-evaluation of their knowledge of phishing risks and anti-phishing solutions. Question 8 was designed to be the dependent variable that measures the subjects' acceptance of and intention to use anti-phishing technology. Additional questions on demographics, such as gender, and age group, years of using computers, average Internet usage, were included in the survey. The eight questions for the survey are as follows:

1. I am aware that online phishing may lead to personal identity theft and financial losses.
2. I am aware of at least one solution to protect against online phishing.
3. https is an important protocol to look for in a secure web address.
4. Before giving my credit card number to an online vendor, I will look for a valid certificate to certify that the vendor web site is secure.
5. I will not likely accept email invitations from an unfamiliar source to give out my personal account information.
6. I am well informed about online phishing risks.
7. I am well informed about anti-phishing technology.
8. I intend to use effective protection technology against online phishing on a regular basis.

The questionnaire was pilot-tested among 15 students. Minor changes in wording were made prior to the formal distribution and administration of the final survey. The survey was distributed to a total of 210 students. The student participation in the survey was declared anonymous and voluntary, and a total of 186 responses were received. 14 responses were eliminated for missing data. A final total of 172 responses were used for data analysis. The next section presents the findings and discussions.

V. FINDINGS AND DISCUSSIONS

Demographic data collected from the subjects included gender, age group, years of using computers, and average Internet usage. The data show that all subjects had at least two years of experience of using computers. Over 85% of the subjects use the Internet between 1 and 6 hours per day;

and over 78% of the subjects have used the Internet for four or more years. The age of the subjects falls between 18 and 56, including 55.7% in age 18-21, 32.4% in age 22-30, 8.9% in age 31-40, 2.3% in age 41-50, and 0.7% above the age of 50. 55% of the subjects were female while 46% were male. The data are close to the general demographics of the student population at the surveyed institution.

SPSS version 18.0 for Windows was used for statistical analysis of the data collected. The reliability analysis of the survey instrument and the Pearson correlations among the eight variables are presented in Table 1 and Table 2 below. The seven independent variables include four Direct Assessment questions (coded as DA1, DA2, DA3, and DA4 represented by questions 1-4 in the survey) and three Indirect Self-assessment questions (coded as SA1, SA2, and SA3 represented by questions 5-7 in the survey). ITN stands for the Intention to Use, which is the dependent variable represented by question 8 in the survey. The valid responses collected were entered into SPSS for reliability and correlations analysis. The analysis reports are presented in TABLE 1 and TABLE 2. The Cronbach's Alpha coefficient is an effective measure of internal consistency reliability, and coefficient values over 0.80 indicate good internal consistency reliability [37]. The Cronbach Alpha values for the four construct items in this study, as shown in Table 1, are all above 0.80 among the eight variables. Therefore, the measures used in this study are considered to have good internal consistency reliability. Discriminant validity of measures is achieved if correlations between any pair of latent constructs are significantly less than 1.00 [38]. The Pearson correlations among the variables shown in Table 2 are significantly less than 1.00. Thus, the measures in this study have demonstrated considerable discriminant validity. In addition, the pilot test of the survey instrument and necessary revisions made in the wording of the survey questions were also helpful in improving the content validity of the questionnaire.

TABLE 1: RELIABILITY STATISTICS

Cronbach's Alpha	Cronbach's Alpha Based on Standardized Items	N of Items
.803	.803	8

TABLE 2: PEARSON CORRELATIONS

	DA1	DA2	DA3	DA4	SA1	SA2	SA3	ITN
DA1	1	.167*	.337**	.274**	.220*	.277**	.312**	.343*
DA2	.167*	1	.319**	.394**	.372**	.208**	.293**	.349**
DA3	.337**	.319**	1	.552**	.325**	.349**	.420**	.544**
DA4	.274**	.394**	.552**	1	.262*	.310**	.305**	.354**
SA1	.220*	.372**	.325**	.262*	1	.312**	.332**	.260*
SA2	.277**	.208**	.349**	.310**	.312**	1	.263*	.292**
SA3	.312**	.293**	.420**	.305**	.332**	.263*	1	.554**
ITN	.343*	.349**	.544**	.354**	.260*	.292**	.554**	1

*. Correlation is significant at the 0.05 level (two-tailed).
**. Correlation is significant at the 0.01 level (two-tailed).

The Pearson correlation results shown in Table 2 indicate support for the research model of this study. The Pearson correlation coefficient is appropriate for interval-scaled measures [37]. The results in the correlations matrix in Table 2 indicate a significant positive relationship between end users' cybersecurity knowledge (via direct assessment and indirect self-assessment) and their intention to use cybersecurity technology solutions. The results also show significant and positive correlations between the Direct Assessment variables and the Indirect Self-assessment variables, which indicates support for the cybersecurity knowledge assessment construct in the research model proposed in this study.

VI. CONCLUSION

This study focuses on the relationship between assessment of users' knowledge of cybersecurity risks and solutions (i.e., phishing risks and anti-phishing solutions) and their attitude and intention toward adoption and use of cybersecurity solutions. Based on the anti-phishing scenario, a novel and simplified technology acceptance model was proposed and tested using a survey study including independent variables of both direct assessment and indirect self-assessment of cybersecurity knowledge. The findings indicate support for the proposition that users' cybersecurity (i.e., phishing) knowledge is positively related to their attitude and intention toward adopting and using cybersecurity (anti-phishing) solutions. The research data also indicate a positive correlation between the direct assessment method and the indirect self-assessment method in evaluating users' cybersecurity knowledge. This correlation may point to the valid practice of including both direct and indirect assessment methods in cybersecurity knowledge training.

User acceptance and adoptions of cybersecurity technology solutions is an emerging and significant area for researchers and the business community. This study contributes a new model of cybersecurity knowledge assessment and user acceptance of cybersecurity solutions. The study also adds valuable data to the study of online phishing risks and user acceptance of anti-phishing solutions. This study has some important practical implications as well. Cybersecurity technology solution vendors need to heed user knowledge level in marketing their products and services. Improving user training and knowledge level may lead to higher levels of acceptance and adoptions of cybersecurity solutions. It is also an opportunity and social responsibility for educational institutions to provide more effective cybersecurity (i.e., anti-phishing) programs and courses to improve user knowledge of cybersecurity risks and cybersecurity technology solutions. In terms of assessing cybersecurity knowledge, the study suggests that direct and objective assessment method can be as equally effective as indirect self-assessment method. Therefore, cybersecurity training programs may consider using both direct assessment and indirect assessment questions in evaluating users' knowledge and competency.

There are promising follow-up research opportunities to pursue this study further. This study focused on the effect of the direct and indirect cybersecurity knowledge variables on user intention to use cybersecurity solutions using samples of general users. Further studies can be conducted among users of different levels of cybersecurity experience and include additional variables, such as specific variables on user experience in online security risks and resolutions as well as effectiveness in communication of cybersecurity risks and solutions. This study is based on the specific area of phishing risks and anti-phishing solutions. Future studies can be done on other cybersecurity topics, such as malware risks and anti-malware solutions, botnets, or the emerging cloud security risks and solutions. In addition, using a larger sample size and a more comprehensive assessment process than those used in this study may produce more informative and conclusive data.

REFERENCES

[1] Phishing Activity Trends Report (2nd Qtr 2012), Published by Anti-phishing Working Group (APWG) at http://www.apwg.org, retrieved: April, 2013.

[2] A. Savvas, "Spear phishing the main email attachment threat," from https://www.networkworld.com/news/2012/112912-39spear-phishing39-the-main-email-264621.html, retrieved: April, 2013.

[3] 2010/2011 Computer Crime and Security Survey, Published by Computer Security Insitute (CSI).

[4] Y. Lee and K. A. Kozar, "An empirical investigation of anti-spyware software adoption: A multi-theoretical perspective," Information & Management, vol. 48, 2008, pp. 109-119.

[5] Q. Yeh and A. J. Chang, "Threats and countermeasures for information system security: A cross-industry study," Information & Management, vol. 44, 2007, pp. 480-491.

[6] Cybersecurity Human Capital (November 2011). Published by United States Government Accountability Office (GAO) at http://www.gao.gov/products/GAO-12-8, retrieved: April, 2013.

[7] T. H. Davenport, D. De Long, and M. Beers, "Successful knowledge management," Sloan Management Review, vol. 39, 1998, pp. 43-57.

[8] S. M. Furnell, P. Bryant, and A. D. Phippen, "Assessing the security perceptions of personal Internet users," Computers & Security, vol. 26, 2007, pp. 410-417.

[9] L. F. Cranor, "Can users control online behavioral advertising effectively?", IEEE Security & Privacy, vol. 10, no. 2, March/April 2012, pp. 93-96.

[10] P. G. Kelley, L. F. Cranor, and N. Sadeh, "Privacy as part of the app decision-making process," CHI 2013, February 2013, pp. 1-11.

[11] K.Desouza,"Facilitating tacit knowledge exchange," Communications of the ACM, vol. 46, June 2003, pp. 85-89.

[12] M. Fishbein and I. Ajzen, Belief, Attitude, Intention, and Behavior: An Introduction to Theory and Research. Reading, MA: Addison-Wesley, 1975.

[13] B. Suh and I. Han, "The impact of customer trust and perception of security control on the acceptance of electronic commerce," International Journal of Electronic Commerce, vol. 7, Spring 2003, pp. 135-161.

[14] J. Jiang, C. Chen, and C. Wang, "Knowledge and trust in e-consumers' online shopping behavior," International Symposium on Electronic Commerce and Security, 2008, pp. 652-656, doi: 10.1109/ISECS.2008.117.

[15] A. Bhatnagar, S. Misra, and H.R. Rao, "On risk, convenience, and internet shopping behavior," Communications of the ACM, vol. 43, no. 11, 2000, pp. 98-105.

[16] A. D. Miyazaki and A. Fernandez, "Consumer perceptions of privacy and security risks for online shopping," The Journal of Consumer Affairs, vol. 35, no. 1, 2001, pp. 27-44.

[17] W. D. Salisbury, R. A. Pearson, A. W. Pearson, and D. W. Miller, "Perceived security and World Wide Web purchase intention," Industrial Management & Data Systems, vol. 101, no.4, pp. 165-176.

[18] P. A. Pavlou, "Consumer acceptance of electronic commerce: integrating trust and risk with the technology acceptance model," International Journal of Electronic Commerce, vol. 7, no. 3, 2003, pp. 69-103.

[19] G. R. Milne, A. J. Rohm, and S. Bahl, "Consumers' protection of online privacy and identity," The Journal of Consumer Affairs, vol. 38, no.2, pp. 217-232.

[20] T. Dinev and Q. Hu, "The centrality of awareness in the formation of user behavioral intention toward protective information technologies," Journal of the Association for Information Systems, vol. 8, July 2007, pp. 386-408.

[21] J. Tsai, L. Cranor, S. Egelman, and A. Acqusiti, "The effect of online privacy information on purchasing behavior: An experimental study," Proceedings of the Twenty Eighth International Conference on Information Systems, Montreal, Canada, 2007, pp. 1-17.

[22] P. Slovic, B. Fischhoff, and S. Lichtenstein, "Why study risk perception?" Risk Analysis, vol. 2, no. 2, 1982, pp. 83-93.

[23] B. Fischhoff, P. Slovic, and S. Lichtenstein, "How safe is safe enough? A psychometric study of attitudes towards technological risks and benefits," Policy Sciences, vol. 9, no. 2, 1978, pp. 127-152.

[24] P. Slovic, "Perception of risk," *Science*, no. 236, 1987, pp. 280-285.

[25] E. A. Nyshadham and M. Ugbaja, "A study of ecommerce risk perceptions among b2c consumers: A two country study," Proceedings of the 19th Bled eConference, Bled, Slovenia, 2006.

[26] I. J. Gabriel and E. Nyshadham, "A cognitive map of people's online risk perceptions and attitudes: An empirical study," Proceedings of the 41st Annual Hawaii International Conference on Systems Sciences, January 2008, pp. 274-283.

[27] I. Ajzen, Attitudes, Personality, and Behavior, Chicago, IL: The Dorsey Press, 1988.

[28] Q. Hu and T. Dinev, "Is spyware an Internet nuisance or public menace?" Communications of the ACM, vol. 48, August 2005, pp. 61-66.

[29] T. Chenoweth, R. Minch, and T. Gattiker, "Application of protection motivation theory to adoption of protective technologies," Proceedings of the 42nd Hawaii International Conference on System Sciences, January 2009, pp. 1-10.

[30] P. Wang, "Information security knowledge and behavior: An adapted model of technology acceptance," 2nd International Conference on Educational Technology and Computer (ICETC), June 2010, pp. 364-367. DOI: 10.1109/ICETC.2010.5529366.

[31] P. Wang and E. A. Nyshadham, "Knowledge of online security risks and consumer decision making: An experimental study," Proceedings of the 44th Hawaii International Conference on System Sciences, January 2010, pp. 1-10.

[32] T. H. Davenport and L. Prusak, Working Knowledge: How Organizations Manage What They Know. Cambridge, MA: Harvard Business School Press, 1998.

[33] L. Suskie, Assessing Student Learning: A Common Sense Guide (2nd ed.). Hoboken, NJ: Wiley, John & Sons, Inc., 2009.

[34] P. Anderson, J. W. Merhout, J. Bernamati, and T. M. Rajkumar, "Are student self-assessments a valid proxy for direct assessments in information systems programs?" Proceedings of the Sixteenth Americas Conference on Information Systems, August 2010, pp. 1-8.

[35] F. D. Davis Jr., A Technology Acceptance Model for Empirically Testing New End-user Information Systems: Theory and Results. Ph.D. dissertation, Massachusetts Institute of Technology, Sloan School of Management, 1986.

[36] S. Grenier, A. Barrette, and R. Ladouceur, “Intolerance of uncertainty and intolerance of ambiguity: Similarities and differences,” Personality and Individual Differences, vol. 39, 2005, pp. 593-600.

[37] U. Sekaran, Research Methods For Business: A Skill Building Approach, 4th ed. Hoboken, NJ: John Wiley & Sons, Inc., 2003.

[38] M. Boudreau, D. Gefen, and D.W. Straub, “Validation in IS research: A state-of-the-art assessment” MIS Quarterly, vol. 25, 2001, pp. 1-16.

The Effect of Destination Linked Feature Selection In Real-Time Network Intrusion Detection

Phiwa Mzila
Modeling and Digital Science, Information Security
Council for Science and Industrial Research (CSIR)
Pretoria, South Africa
pmzila@csir.co.za

Eric Dube
Modeling and Digital Science, Information Security
Council for Science and Industrial Research (CSIR)
Pretoria, South Africa
Edube1@csir.co.za

***Abstract*—As internet usage rapidly increases in both private and corporate sectors, the study of network intrusion detection is continuously becoming more relevant and has thus been evolving substantially in recent years. One of the most interesting techniques in the network intrusion detection system (NIDS) is the feature selection technique. The ability of NIDS to accurately identify intrusion from the network traffic relies heavily on feature selection, which describes the pattern of the network packets. The objective of this paper is to eliminate unnecessary features from the dataset, namely destination linked features of the network packet, and train a classification model on the remaining features using a k-Nearest Neighbor (k-NN) classifier. Elimination of the insignificant features leads to a simplified problem and may enhance detection rate, which is itself a problem in network intrusion detection system. Furthermore, removal of specifically the destination linked features will allow the trained model to be capable of identifying the attack/intrusion in real-time before it reaches its destination. To evaluate the accuracy of this method, we compare the results of our model trained without destination linked features to the same model trained with features incorporating destination linked features. The results show a similar detection rate for both trained models, but our model has a distinct advantage in that it treats the entire transaction in real-time.**

Keywords-feature selection; pattern recognition;data mining intrusion detection

I. Introduction

The internet has become a standard communication tool in the modern world. It plays an essential role in running most successful businesses. However, together with the advantages the internet brings, there is also a substantial disadvantage. It exposes both valuable and confidential information to high risks of intrusion and cyber-attacks. A vast amount of research has been conducted in order to prevent such attacks, and a multitude of systems have been designed or proposed in recent decades. Most intrusion detection systems are based on signatures that are developed by manual coding of expert knowledge. These systems match activity on the system being monitored to known signatures of attack. The major problem with this approach is that these network intrusion detection systems fail to generalize to detect new attacks or attacks without known signatures.

Recently, there has been an increased interest in data mining based approaches to build detection techniques for network intrusion detection systems. These techniques are constructed from models that are trained by both known attacks and normal behavior in order to detect unknown attacks. [1][2][3] describe some of the effective data mining techniques that have been developed recently for detecting intrusions in computer networks. However, successful data mining techniques are not sufficient to create effective network intrusion detection systems (NIDS). Although data mining techniques are successful in evaluating the detection rate of the NIDS, there are still some difficulties involved in the implementation. These difficulties can be grouped into three general categories: accuracy (e.g., detection rate), efficiency, and usability.

Another concern about the NIDS is that it should operate in real-time. Currently even existing so called real-time intrusion detection systems have been modelled with data processed off-line. Any NIDS that has been modelled using a dataset with features such as duration, destination port, totalDestinationBytes, totalDestinationPackets, destination, and even stopTime are in fact not real-time intrusion detection systems. Attacks should be prevented or identified before reaching its destination. An effective NIDS should work in real-time, as intrusions take place, to avoid compromising security.

Elimination of the insignificant and/or meaningless inputs leads to a simplification of the problem, as well as faster and more accurate detection results. Feature selection is therefore an important issue in intrusion detection. Any intrusion detection system has some inherent requirements. Its prime purpose is to detect as many attacks as possible with minimum number of false alarms, i.e., the system must be accurate in detecting attacks. However, an accurate system that cannot handle large amounts of network traffic and is slow in decision making will not fulfil the purpose of an NIDS. Data mining techniques like pattern recognition, data reduction, data classification, and feature selection techniques thus play an important role in the design of an NIDS.

Feature selection is one of the data preprocessing techniques used before classification in an NIDS [4]. Its purpose is to improve the classification detection accuracy through the removal of irrelevant, noisy and redundant features. Feature selection methods generate a new set of features by selecting only a subset of the original features [5].

There are two main feature selection methods: filter methods [6] and wrapper methods [7]. Filter methods evaluate the relevance of the features depending on the general characteristics of the data, without using any machine learning algorithm to select the new set of features [8]. Frequently used filter methods include Information Gain (IG) [9] and Chi-square [10]. Wrapper methods use the classification performance of a machine learning algorithm as the evaluation criterion to select the set of best features [11]. Wrapper methods include the Particle Swarm Optimization (PSO) algorithm [12] and Genetic Algorithm (GA) [13].

For developing intrusion detection systems, a large amount of traffic data is necessary, which must be collected in advance for analysis by the misuse detection or anomaly detection approaches. Based on the collected network audit trail, misuse detection techniques specify well defined attack signatures and anomaly detection techniques establish acceptable usage profiles to differentiate intrusions and normal activities from a future network traffic data stream. However, there are three major problems in the collected network traffic database: problem of irrelevant and redundant features, problem of uncertainty, and problem of ambiguity.

In this paper, we present a real-time feature selection method for an NIDS which avoids the problem of irrelevant features. The model is trained with a dataset excluding all destination linked features. Hence, the selection of features from the raw dataset constitutes a vital step in the process. The destination and arrival time (duration) of the attack are considered unimportant features in constructing a model that can detect an attack before it arrives at its destination.

For evaluating the detection performance of proposed feature selection method, we compare our results with Soft Computing Paradigms Feature Selection (SCPFS) [14], Correlation Based Feature Selection (CFS) [15] and Fast Correlation-Based Filter (FCBF) [16].

This paper is organized as follows. Section 2 describes the related work. Section 3 presents the methodology which includes the description of the dataset and tools used in this paper followed by a proposed framework in Section 4. We then demonstrate the experimental results in Section 5. Finally, we conclude our work and discuss future recommendations.

II. Related Work

A number of systems aimed at improving network intrusion detection have been developed over the years. In [17], an anomaly network intrusion detection system was proposed using a Particle Swarm Optimization (PSO) feature selection method and Information Entropy Minimization (IEM) discretization with Hidden Naive Bays (HNB) classifier. The effectiveness of the proposed network IDS was evaluated by conducting several experiments on NSL-KDD network intrusion dataset. The results showed that the proposed PSO-Discritize-HNB IDS increases the accuracy and decreases the detection time.

Most of the related work in anomaly detection uses Self-Learning Artificial Neural Networks (ANN) as in HyperView [18]. The system’s normal traffic is fed to an ANN, which subsequently learns the pattern of normal traffic. The new traffic, including possible attacks, is then applied to the ANN and the output is used to form the intrusion detection decision. Other systems utilize descriptive statistics by collecting uni-modal statistics from certain system parameters into a profile, after which a distance vector is constructed for the observed traffic and the profile. If the distance is great enough the system raises the alarm. Examples of these systems are NIDES [19], EMERALD [20] and Haystack [21].

A system developed by Girardin [22] used multiple self-organizing maps for intrusion detection. A collection of more specialized maps was used to process network traffic for each layered protocol separately. Girardin suggested that each neural network become a specialist, trained to recognize the normal activity of a single protocol. Another approach that differs from anomaly detection and misuse detection considers human factors to support the exploration of network traffic. Girardin used self-organizing maps to project the network events onto a space appropriate for visualization, and achieved their exploration using a map metaphor

Chen et al [23] used Rough Set Theory (RST) and Support Vector Machines (SVM) to detect intrusions. Initially, RST was used to preprocess the data and reduce the dimensions. Later, the features selected by RST were sent to an SVM model to learn and test. This method proved to be effective and also decreased the space density of data. The SVM is one of the most successful classification algorithms in the data mining area [24].

Statistical techniques usually assume an underlying distribution of data and require the elimination of data instances containing noise. Statistical methods are therefore computationally intensive but can be applied successfully to analyse the data [25]. Statistical methods are widely used to build a behaviour based IDS. The behavior of the system is measured by a number of variables sampled over time such as the resource usage duration, the number of processors, and memory disk resources consumed during that session. The model keeps averages of all the variables and detects whether thresholds are exceeded based on the standard deviation of the variable.

Al-Subaie et al [26] used Hidden Markov Models over Neural Networks in anomaly intrusion detection to classify normal network activity and attacks using a large training dataset. The approach was evaluated by analysing how it affected the classification results. Amor et al [27] designed a real time IDS using Naïve Bayes and Decision Trees and the results showed that the Naive Bayes gives higher detection speed and detection rate than the Decision Trees. Authors in [28] and [29] proposed a hybrid intelligent system using Decision Trees (DT), SVM and Fuzzy SVM for anomaly detection (unknown or new attacks). The results showed that the hybrid DT–SVM approach improved the performance for all the classes when compared to an SVM approach.

Most of these approaches address the feature selection process based mostly in random feature reduction which is not sufficient for intrusion detection in real-time.

III. METHODOLOGY

A. Dataset Description

To evaluate the performance, namely the detection rate and accuracy of the proposed feature selection method of real-time IDS system, we used the Information Security Centre of Excellence (ISCX 2012) dataset [30]. The ISCX 2012 dataset has been created by security researchers at ISCX including Ali Shiravi, Hadi Shiravi, and Mahbod Tavallaee. The dataset was designed to aid research efforts in developing, testing and evaluating algorithms for intrusion detection and anomaly detection. This came about due to the fact that anomaly-based approaches in particular suffer from inaccurate evaluation, comparison, and deployment which originate from the scarcity of adequate datasets. Many such datasets are internal and cannot be shared due to privacy issues, others are heavily anonymized and do not reflect current trends, or they lack certain statistical characteristics. At ISCX, a systematic approach to generate the required datasets was introduced to address this need. The data consists of 17 features shown in the Table I below and the "Tag" value indicates whether the flow is normal or an attack.

TABLE 1. LIST OF FEATURES COLLECTED

appName	sourceTCPFlagsDescription
totalSourceBytes	destinationTCPFlagsDescription
totalDestinationBytes	source
totalDestinationPackets	protocolName
totalSourcePackets	sourcePort
sourcePayloadAsBase64	destination
destinationPayloadAsBase64	destinationPort
direction	startDateTime
Tag	stopDateTime

B. RapidMiner

To implement our method we used RapidMiner [30]. RapidMiner is an international open-source data mining framework. It enables users to model complex knowledge discovery processes as it supports nested operator chains. There are several reasons which made RapidMiner the data mining tool of choice. RapidMiner can function on-line on a given data set, which was necessary for this application. It has many data loading, modeling, preprocessing and visualization methods. This avoids the need for preprocessing the data sets. It also enables visualization of the results. It has an easy to use yet robust graphical user interface that facilitates the modeling of different complex processes. It is also modular, which allows the use of additional functionalities, for example, the distance measures used for the anomaly detection operators. Finally it is easily extensible.

C. k-Nearest Neighbor

Our choice for the classification technique was the k-Nearest Neighbor algorithm since it easy to implement and produces better results. It is based on learning by analogy, that is, by comparing a given test example with training examples that are similar to it. The training examples are described by n attributes. Each example represents a point in an n-dimensional space. When given an unknown example, a k-nearest neighbor algorithm searches the pattern space for the k training examples that are closest to the unknown example. These k training examples are the k "nearest neighbors" of the unknown example. "Closeness" is defined in terms of a distance metric, such as the Euclidean distance. The k-nearest neighbor algorithm is amongst the simplest of all machine learning algorithms: an example is classified by a majority vote of its neighbors, with the example being assigned to the class most common amongst its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the example is simply assigned to the class of its nearest neighbor. The same method can be used for regression, by simply assigning the label value for the example to be the average of the values of its k nearest neighbors. It can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. The neighbors are taken from a set of examples for which the correct classification (or, in the case of regression, the value of the label) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required. The basic k-Nearest Neighbor algorithm is composed of two steps: Find the k training examples that are closest to the unseen example. Take the most commonly occurring classification for these k examples (or, in the case of regression, take the average of these k label values).

D. Experimental Conditions

There were two phases in the experiment. In the first phase we evaluated the performance of our method (off-line classification), using full feature sets of the network traffic dataset. In the second phase (real-time) only selected features, excluding all destination linked features of the network traffic datasets, were used.

IV. PROPOSED FRAMEWORK

Fig. 1 shows the overall framework of the process involved in the proposed feature selection model for an

NIDS. We adopted the TCM-KNN (Transductive Confidence Machines for K-Nearest Neighbors) [31]

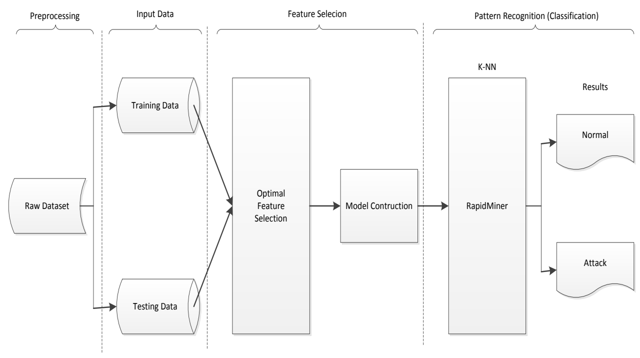


Figure 1. Architecture of the model.

The framework includes four phases: preprocessing, input data, feature selection, and classification.

- Preprocessing: The raw dataset is collected and preprocessed for easy interfacing with RapidMiner format. Each instance of this data is labelled as either an attack or normal.
- Input Data: As input, the preprocessed dataset is split into training and testing datasets. The training dataset is used for training the model with the pattern of both attack and normal network traffic. The testing dataset is unlabelled during the preprocessing phase, and its purpose is to evaluate how well the model is trained for the unknown traffic to detect if it is an attack or normal.
- Feature Selection: Feature selection occurs during model construction. This is the phase where we select features that can construct a model for real-time intrusion detection. All features which are destination linked such as destination port, duration and stop time are deselected in this phase.
- Classification: A well trained model using k-Nearest Neighbour algorithm takes in a testing dataset without label and classifies whether each entry's pattern is closer to those derived from the normal or attack entries of the training dataset.

V. TCM-KNN Algorithm

RapidMiner has the capability to compute the confidence using algorithmic randomness theory which was introduced by Transductive Confidence Machines (TCM) [32]. Unlike traditional methods in data mining, transduction can offer measures of reliability to individual points, and uses very broad assumptions except for the main assumption (the training as well as new (unlabelled) points are independently and identically distributed). The calculated p-value serves as a measure of how well the data supports or rejects a null hypothesis (that the point belongs to a certain class). The smaller the p-value, the greater the evidence against the null hypothesis (i.e., the point is an outlier with respect to the current available classes). Users of transduction as a test of confidence have approximated a universal test for randomness (which is in its general form, non-computable) by using a p-value function called strangeness measure [33]. The concept is that the strangeness measure corresponds to the uncertainty of the point being measured with respect to all the other labelled points of a class. Imagine we have an intrusion detection training set $\{(x_1,x_1),..., (x^n, y^n)\}$ of n elements, where $\{X_i = x^1_i, x^2_i, \ldots x^n_i\}$ is the set of feature values (such as the connection duration time, the packet length, etc.) extracted from the raw network packet (or network flow such as TCP flow) for point i, and y_i is the classification for point i, taking values from a finite set of possible classifications (such as normal, attack, intrusion, etc.), which we identify as {1,2,3,..., c} . A test set of s points similar to the ones in the training set is used. The goal is to assign to every test point one of the possible classifications. For every classification confidence measures are also assigned.

VI. Experimental Results

To evaluate the effect of our method of eliminating destination linked features, a dataset of 2000 labelled instances, either as an attack or normal traffic, was used. The k-NN classifier was applied in a cross validation for performance evaluation using RapidMiner operators tool. Cross-validation is a standard statistical method to estimate the generalization error of a predictive model, where each subset of the data is used as a test set with the other subsets forming the training set. We divided the dataset into k=10 equal-sized subsets for k-fold cross-validation. The following procedure was repeated for each subset: Our model was built using the other (k-1) subsets as the training set and its performance was evaluated on the current subset. This means that each subset was used for testing exactly once. The result is the average of the performances obtained from the k=10 rounds.

To determine the detection rate, we used normal evaluation equations with standard measurements, *detection rate* (*DR*) and *false positive rate (FPR*) as described in equation 1 and 2, respectively.

$$DR = \frac{TP}{TP+FN} \tag{1}$$

$$FPR = \frac{FP}{TN+FP} \tag{2}$$

The denotations of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) are defined as follows:

- True Positives (TP): The number of malicious records that are correctly identified.
- True Negatives (TN): The number of legitimate records that are correctly classified.
- False Positives (FP): The number of records that were incorrectly identified as attacks when they are in fact legitimate activities.

- False Negatives (FN): The number of records that were incorrectly classified as legitimate activities when in fact they are malicious.

Tables 2 and 3 below show the performance vector of both the full feature set and reduced feature set (destination linked features removed) respectively. The performance vector comprises of different parameters, which are: accuracy, precision, recall, confusion matrix and class prediction.

TABLE 2.PERFOMANCE VECTOR ON FULL FEATURE SET

```
PerformanceVector:

Accuracy: 92.05% +/- 2.04% (mikro: 92.05%)
Precision: 82.14% +/- 5.06% (mikro: 81.78%) (Positive class: Attack)
Recall: 93.86% +/- 3.80% (mikro: 93.88%) (Positive class: Attack)

ConfusionMatrix:
True:       Normal      Attack
Normal:     1289        36
Attack:     123         552

Class Prediction: Normal: 97.28%
Class Prediction: Attack: 81.78%
```

TABLE 3. PERFOMANCE VECTOR ON REDUCED FEATURE SET

```
PerformanceVector:

Accuracy: 90.70% +/- 1.49% (mikro: 90.70%)
Precision: 79.26% +/- 4.09% (mikro: 78.96%) (positive class: Attack)
Recall: 93.18% +/- 3.99% (mikro: 93.20%) (positive class: Attack)

ConfusionMatrix:
True:       Normal      Attack
Normal:     1266        40
Attack:     146         548

Class Prediction: Normal: 96.94%
Class Prediction: Attack: 78.96%
```

We compare the DR and FPR rate results of our method with the results of three other methods which are Soft Computing Paradigms Feature Selection (SCPFS), Correlation Based Feature Selection (CFS) and Fast Correlation-Based Filter (FCBF). The results are depicted in Table 4.

The result when using the reduced feature set is lower than the accuracy of the full set of features. The reason for this is the significantly reduced number of features from 17 to 10. Reduction in the number of features is not our primary objective in this work but we do not rule out the possibility that it played a huge role in achieving better results for our method. Our objective is to eliminate precisely all destination linked features of the dataset for training a model that can detect attacks in real-time.

TABLE 4. RESULTS COMPARISON OF DR AND FPR PERFOMED ON K-NN USING FULL FEATURE SET AND REDUCED FEATURE SET

Parameter	**Full Set**	**Proposed Method**	**SCPFS**	**CFS**	**FCBF**
DR	70.01	70.00	83.21	12.20	6.87
FPR	0.23	0.22	9.59	0.25	0.13
Accuracy	**92.05**	**90.70**	-	-	-

In Table 4, we have included the accuracy of our method based on the results obtained during our experiments. These results are affected by the significantly decreased number of features participating in the experiment and the size of the dataset used. With a larger dataset the outcome is likely to be different between the full feature set and reduced feature set results. Detection rate for proposed method is lower than SCPFS possibly due to the reduced feature set, but higher than CFS and FCBF. An ideal network intrusion detection system must have lowest false positive rate and the proposed method is lower than SCPFS but higher than CFS and FCBF. There was no significant difference between DR and FPR for the two feature sets, but the main advantage of the proposed method is the real-time application.

VII. CONCLUSION AND FUTURE WORK

In this paper, we proposed a feature selection model for a real-time intrusion detection system. We created a model based on de-selection of destination linked features of the network traffic with the aim of removing features that are irrelevant for a real-time application. The real time intrusion detection system should be able to detect if the traffic instance is an attack before reaching its destination, hence all destination linked features become unnecessary. We then compared our results with the model constructed using a full set of features of the network traffic. The approach using a full feature set of the network traffic represents an off-line intrusion detection scenario while the approach with de-selection of destination linked features represents the real-time intrusion detection scenario. Only a small difference in accuracy were found between the two feature sets, indicating that the proposed reduced feature set did not significantly reduce the accuracy, while still having the main advantage of its suitability for real-time use. Our future work is to develop a real-time network intrusion detection system that is able to detect any intrusion pattern in the network before an attack reaches its target destination using a larger dataset, as well as to improve the accuracy.

REFERENCES

[1] I.T. Jolliffe, “Principal component analysis”, New York Springer-Verlag, 1986.

[2] J. S. Chandrasekaran, B.S. Manjunath, Y.F. Wang, J. Winkeler and H. Zhang, “An eigenspace update algorithm for image

analysis," Graphical Models and Image Processing, 59(5), pp.321-332, 1997.

[3] J. Winkeler, B.S. Manjunath and S. Chandrasekaran, "Subset selection for active object recognition," In CVPR, volume 2, IEEE Computer Society Press, pp.511-516, 1999.

[4] M. Ben-Bassat, Pattern recognition and reduction of dimensionality, Handbook of Statistics II, vol. 1, North-Holland, Amsterdam, pp.773-791, 1982.

[5] H. Liu and H. Motoda, Feature Extraction, Construction and Selection: A Data Mining Perspective, Kluwer Academic, second printing, Boston, 2001.

[6] H. F. Eid and A. Hassanien, "Improved real-rime discretize network intrusion detection model", Seventh International Conference on Bio-Inspired Computing: Theories and Application (BIC-TA 2012), December 14-16, Gwalior, India, 2012.

[7] L. Yu and H. Liu, "Feature selection for high-dimensional data: a fast correlation based filter solution", In Proc. of the Twentieth International Conference on Machine Learning, pp. 856-863, 2003.

[8] Y. Kim, W. Street and F. Menczer, "Feature selection for unsupervised learning via evolutionary search", In Proc. of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 365-369, 2000.

[9] H. Almuallim and T.G. Dietterich, Learning Boolean Concepts in the Presence of Many Irrelevant Features, Artificial Intelligence, vol. 69, pp. 279-305, 1994.

[10] X. Jin, A. Xu, R. Bie and P. Guo, Machine learning techniques and chi-square feature selection for cancer classification using SAGE gene expression profiles, Lecture Notes in Computer Science, 3916, DOI: 10.1007/1169173011, pp. 106-115, 2006.

[11] Y. Kim, W. Street and F. Menczer, "Feature selection for unsupervised learning via evolutionary search", In Proc. of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 365-369, 2000.

[12] L. Chuang, C. Ke and C. Yang, "A hybrid both filter and wrapper feature selection method for microarray classification", In Proc. of the International Multi Conference of Engineers and Computer Scientists (IMECS), Hong Kong, vol. 1, pp. 19-21, 2008 .

[13] C. Yang, L. Chuang and C. Hong Yang, "IG-GA: A hybrid filter/wrapper method for feature selection of microarray data", Journal of Medical and Biological Engineering, vol. 30, pp. 23-28, 2009.

[14] T. S. Chou, K. K. Yen, and J. Luo, "Network intrusion detection design using feature selection of soft computing paradigms", International Journal of Information and Mathematical Sciences 4:3, 2008.

[15] M. Hall, "Correlation based feature selection for machine learning" Doctoral Dissertation, The University of Waikato, Department of Computer Science, 1999.

[16] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in Proceedings of The Twentieth International Conference on Machine Leaning, Washington, D.C., August, pp. 856-863, 2003.

[17] A. Ahmed Elngar1, A, Dowlat, A. El Mohamed and F. Fayed, "A real-time network intrusion detection system with high accuracy", Information & Computer science Faculty, Sinai University, El-Arish, 2012.

[18] H. Debar, M. Becker, D. Siboni, "A neural network component for an intrusion detection system". Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, 1992.

[19] P. Porras, P. Neumann, "EMERALD: Event monitoring enabling responses to anomalous live disturbances", Proceedings of the 20th National Information Systems Security Conference, Baltimore, Maryland, 1997.

[20] S. Smaha, "Haystack: An intrusion detection system" Proceedings of the IEEE forth Aerospace Computer Security Applications Conference, Orlando, Florida, 1988.

[21] B. Rhodes, J. Mahaffey, and J. Cannady, "Multiple Self-Organizing Maps for intrusion detection". Proceedings of the NISSC, 2000.

[22] L. Girardin, "An eye on network intruder- administrator shootouts". Proceedings of the Workshop on Intrusion Detection and Network Monitoring, Santa Clara, CA, USA, April 9-12, 1999.

[23] R. Chen, K. Cheng, and C. Hsieh, "Using rough set and support vector machine for network intrusion detect", International Journal of Network Security & Its Applications (IJNSA), 2009.

[24] J.M. Moguerza and Alberto Munoz, "Support vector machines with applications", Statistical Science, vol. 21, No. 3, pp. 322 – 336, 2006.

[25] G. M. Nazer, A. A. L. Selvakumar, "Current intrusion detection techniques in information", European Journal of Scientific Research, EuroJournals Publishing, Inc., pp. 611-624, 2011.

[26] M. Al-Subaie and M. Zulkernine, "Efficacy of Hidden Markov Models over neural networks in anomaly intrusion detection", In 30th Annual International Computer Software and Applications Conference (COMPSAC'06), pp. 325–332, 2006.

[27] B. Amor, S. Benferhat and Z. Elouedi, "Naive Bayes vs. Decision Trees in intrusion detection systems", In SAC '04: Proceedings of the 2004 ACM symposium on applied computing, New York, NY, USA, ACM. ISBN 1-58113-812-1, pp. 420–424, 2004

[28] S. Peddabachigari, A. Abraham, C. Grosanc, and J. Thomas, "Modeling intrusion detection system using hybrid intelligent systems", Journal of Network and Computer Applications, Elsevier Ltd, 2005.

[29] S. Teng, H. Du, N. Wu, W. Zhang, and Jiangyi Su, "A cooperative network intrusion detection based on Fuzzy SVMs", Journal of Networks, vol. 5, pp. 474-483, 2010 .

[30] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection", Computers & Security, vol. 31, Issue 3, ISSN 0167-4048, 10.1016/j.cose.2011.12.012, pp. 357-374, 2012.

[31] Data Mining / Analytic Tools Used Poll". Data Mining / Analytic Tools Used Poll (May 2010). KDnuggets. Retrieved 4 July 2012.

[32] Y. Li, "An effective TCM-KNN scheme for high-speed network anomaly detection", International Journal of Advanced Science and Technology Vol. 24, Chinese Academy of Sciences, Beijing China, 100080, 2010

[33] A. Gammerman, and V. Vovk, "Prediction algorithms and confidence measure based on algorithmic randomness theory". Theoretical Computer Science, pp. 209-217, 2002.

SLOPPI — a Framework for Secure Logging with Privacy Protection and Integrity

Felix von Eye, David Schmitz, and Wolfgang Hommel
Leibniz Supercomputing Centre, Munich Network Management Team, Garching near Munich, Germany
Email:{voneye,schmitz,hommel}@lrz.de

***Abstract*—Secure log file management on, for example, Linux servers typically uses cryptographic message authentication codes (MACs) to ensure the log file's integrity: If an attacker modifies or deletes a log entry, the MAC no longer matches the log file content. However, some privacy and data protection laws, for example in Germany, require the deletion or anonymization of log entries with personal data after a retention period of seven days. Such changes therefore do not constitute an attack. Previous work regarding secure logging does not support this use case adequately. A new log management approach with a focus on both the integrity and the compliance of the resulting log files with additional support for encryption-based confidentiality is presented and discussed.**

Keywords-log file management; secure logging; compliance

I. Introduction

System and application logs are of great value for administrators, e. g., for monitoring, fault management, and forensics. The predominant format for logs is plain text, which is the focus here; for example, the *syslog daemon* on UNIX and Linux systems, as well as applications like the Apache web server, store log entries in line-oriented plain text. Alternatively, proprietary binary or XML-based file formats as well as relational databases may be used; for example, this is used by the Microsoft Windows event log. Independent of the storage format, *secure* logs must fulfill the following basic criteria in practice:

- The log's *integrity* must be ensured: Neither a malicious administrator nor an attacker, who has compromised a system, may be able to delete or modify existing, or insert bogus log entries.
- The log must not violate *compliance* criteria [1]. For example, European data protection laws regulate the retention of personal data, which includes, among many others, user names and IP addresses. These restrictions also apply to log entries according to several German courts' verdicts that motivated our work.
- The *confidentiality* of log entries must be safeguarded; i. e., read access to log entries must be confined to an arbitrary set of users.
- The *availability* of log entries must be made sure of.

In a typical data center or network operations environment, the *integrity* criterion is usually fulfilled by using a trustworthy, central log server: Log entries are not (solely) stored locally on a device, but sent over the network to another machine; therefore, an attacker would have to compromise both this device and the log server before being able to manipulate the log without being detectable. The *compliance* criterion can be fulfilled by deleting old log files, e. g., after the common duration of seven days. As of today, *confidentiality* is typically handled in a per-file manner; for example, UNIX file system permissions are often used to make a log file readable for a specific user or group. Prospective modern logging facilities also enable confidentiality in a per-entry granularity, but are not yet in such wide use. Finally, the *availability* requirement for log files is the same as for other important data and typically ensured by data redundancy, i. e., copies and backups.

Our work is motivated by the large-scale distributed environment of the SASER-SIEGFRIED project (Safe and Secure European Routing) [2], in which more than 50 project partners design and implement network architectures and technologies for secure future networks. The project's goal is to remedy security vulnerabilities of todays IP layer networks in the 2020 timeframe. Thereby, security mechanisms for future networks will be designed based on an analysis of the currently predominant security problems in the IP layer, as well as upcoming issues such as vendor backdoors and traffic anomaly detection. The project focuses on inter-domain routing, and routing decisions are based on security metrics that are part of log entries sent by active network components to central network management systems; therefore, the integrity of this data must be protected, providing a use case that is similar to traditional intra-organizational logfile management applications.

In this paper, the focus is on *integrity* as well as *compliance* and the presentation of a novel, cryptographically enhanced log storage and processing approach that fulfills both criteria with or without a central log server. In the SASER-SIEGFRIED context, a central log server can use the presented framework to proof the logs' integrity so that the security metrics based on data from this log source can be considered reliable. On the other hand, the presented framework can also be used in other scenarios where a central log server may not be applicable or may not be a suitable solution, e. g.:

- Small organizations or small enterprises that do not have the resources, i. e., budget and/or skills, to operate a central log server.
- Machines and devices that do not have permanent network connectivity to a central log server, e. g., sensor networks, mobile devices, or servers that keep radio silence for a purpose, such as honeypots, which must appear to be easy prey for attackers.

- Massively distributed systems with a high rate of log entries that do not need to be correlated on a regular basis, in which a central log server would become a performance bottleneck.

Especially in environments without trustworthy systems, such as a reliable central log server, cryptographic functions are used to make log files more secure. For example, message authentication codes (MACs) can be used to make log files evident of simple modifications: If an attacker manipulates or deletes a log entry, the log file's MAC no longer matches its content, making the attack obvious. However, previous work on secure logging did not account for desired changes to the log entries' content, such as the deletion of old log files. Doing so required the complex re-calculation of MACs and cryptographic hash values, impeding the practical use of these other approaches.

We present a secure logging approach that supports this use case of deleting old log files after an arbitrary retention period without re-keying. This approach uses cryptographic hash functions and asymmetric encryption to implement log entry integrity verification. Fresh cryptographic key material is generated after an arbitrary amount of time, e. g., daily, or when a certain number of log entries have been processed. However, unlike previous approaches, the solution fulfills the compliance criterion by allowing log files to be removed without violating the overall log data integrity and without the need to keep old cryptographic keys or re-calculate MACs for the whole log file. This approach also keeps the log files needed by applications in plain text, so they can, unlike a fully encrypted log file, be processed using any software tools that the administrator is already familiar with.

The presented solution can be extended to also ensure per-entry confidentiality and additional measures can be taken to improve the high availability. However, *integrity* and *compliance* are in the focus of this paper, which is structured as follows: In the next section, the terminology that is used throughout this paper is introduced. In Section III, previous approaches to secure logging and related work are summarized in order to outline the shortcomings that are addressed. The details of our approach are presented in Section IV. The paper closes with a conclusion in Section VI.

II. Terminology

The following terms and symbols are used in this paper with the specified meaning:

- The untrusted device $\mathcal{U}$, e. g., a web server. As a matter of its regular operations, $\mathcal{U}$ produces log data. It could be assumed that $\mathcal{U}$ may be compromised by an attacker and therefore the log data is not guaranteed to be *trustworthy*. However, this approach can be used to ensure the integrity and compliance of log data produced by $\mathcal{U}$, making it *reliable*.
- A trusted machine $\mathcal{T}$. In any related work and also in the presented work there is a need for a separate machine $\mathcal{T} \neq \mathcal{U}$. The working assumption in the related work is that $\mathcal{T}$ is secure, trustworthy, and not under the control of an attacker at any point in time. In the presented approach, $\mathcal{T}$ could be a connected printer, because a place is needed where only one initial key is saved, i. e., printed, without the possibility of intruder access.
- The verifier $\mathcal{V}$. The related work often differentiates $\mathcal{T}$ and $\mathcal{V}$; $\mathcal{V}$ then is only responsible for verifying the integrity and compliance of a log file or log stream. In this case, $\mathcal{T}$ is only used to store the needed keys and $\mathcal{V}$ has not to be as trustworthy as $\mathcal{T}$. Also in this case $\mathcal{T}$ is able to modify any log entry while $\mathcal{V}$ is not.

These symbols are used for cryptographic operations:

- A strong cryptographic hash function H, which has to be a one way function, i. e., a function which is easy to compute but hard to reverse, e. g., `sha-256(`m`)`.
- HMAC$_k(m)$. The message authentication code of the message m using the key k.

In our proposal we do not anticipate, which particular function should be used for cryptographic functions; instead, they should be chosen specifically based on each implementation's security requirements and constraints, such as available processing power and storage overhead.

Furthermore, without loss of generality, the terms *a)* log files, *b)* log entries, and *c)* log messages are discerned. A *log file* is an ordered set of *log entries*. For example, on a typical Linux system, `/var/log/messages` is a text-based log file and each line therein is a log entry; log entries are written in chronological order to this log file. *Log messages* are the payload of *log entries*; typically, log messages are human-readable strings that are created by applications or system/device processes. Besides a log message, a log entry includes metadata, such as a timestamp and information about the log message source.

As shown in Figure 1, the presented approach uses a couple of log files, which are related to each other in the following way. The *master log* L_m is only used twice a day to first generate and to then close a new integrity stream for the so-called *daily log* L_d. This log file L_d is used to minimize the storage needs for the master log L_m, which must never be deleted; otherwise, no complete verification could be performed. L_d is kept as long as necessary and contains a new integrity stream for the *application logs* L_a. These logs, e. g., the `access.log` or the `error.log` by an Apache web server or the `firewall.log` by a local firewall, are re-started from scratch once per day and yield all information generated by the related processes; they are extended by integrity check data. After a specified retention time – seven days in the scenario – these logs, which contain privacy-law protected data, have to be deleted completely because of legal constraints in Germany and various other countries. It is important to mention that a simple deletion would also remove any information about attempted intrusions and other attack sources. This would cover an intruder, who could be detected by analyzing the log files, so the log file should be analyzed periodically before this automated deletion. Other time periods than full days or a rotation that is based,

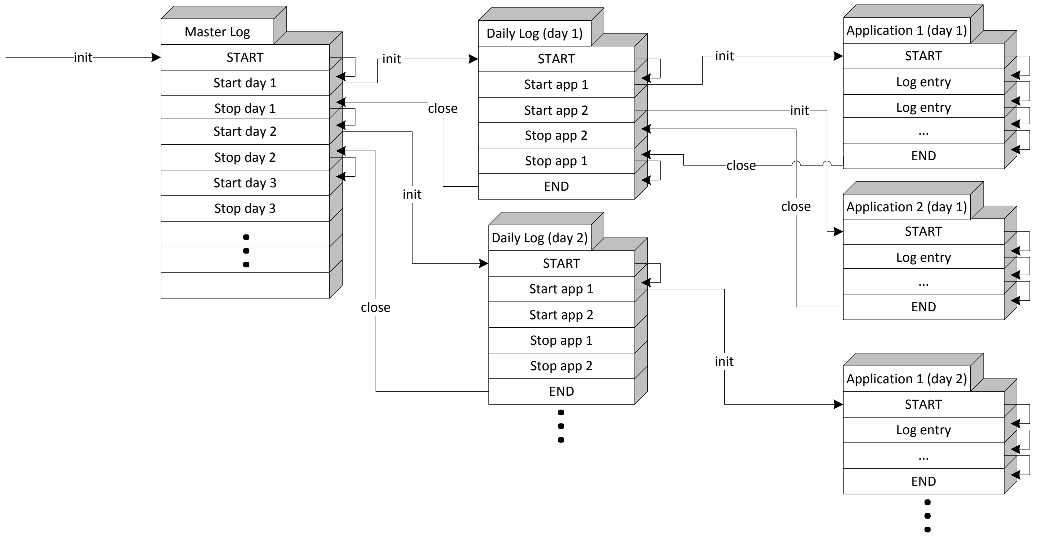


Fig. 1. Overview of all relevant log files.

e. g., on a maximum number of log entries per log file, as well as other deletion periods may be applied – but for the sake of simplicity *daily* logs and a seven day deletion period are used for the remainder of this paper.

III. RELATED WORK

With the exception of Section III-A, none of related work offers a possibility to fulfill compliance as it is not possible to delete logs afterwards. Complementary, the approach summarized in Section III-A does not address the integrity issues.

A. Privacy-enhancing log rotation

Metzger et al. [3] presented a system, which allows privacy-enhancing log rotation. In this work, log entries are deleted by log file rotation after a period of seven days, which is a common retention period in Germany based on several privacy-related verdicts. Based on surveys, Metzger et al. identified more than 200 different types of log entry sources that contain personal information in a typical higher education data center. Although deleting log entries after seven days seems to be a simple solution, the authors discuss the challenges of implementing and enforcing a strict data retention policy in large-scale distributed environments.

B. Forward Integrity

Bellare and Yee [4] introduced the term *Forward Integrity*. This approach is based on the combination of log entries with message authentication codes (MACs). Once a new log file is started, a secret s_0 is generated on $\mathcal{U}$, which has to be sent in a secure way to a trusted $\mathcal{T}$. This secret is necessary to verify the integrity of a log file.

Once the first log entry l_0 is written in the log file, the HMAC of l_0 based on the key s_0 is calculated and also written to the log file. To protect the secret s_0, there is another calculation of $s_1 = H(s_0)$, which is the new secret for the next log entry l_1. To prevent that an attacker can easily create

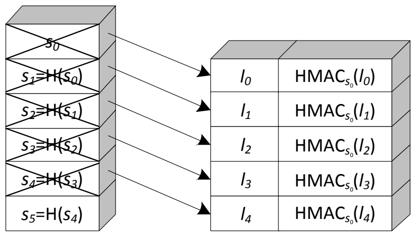


Fig. 2. The basic idea behind Forward Integrity as suggested in [4]

or modify log entries, the old and already used secret key for the MAC function is erased after the calculation securely. Because of the characteristic of one way functions it is not possible for an attacker to derive the previous key backwards in maintainable time. Figure 2 shows the underlying idea.

In their approach, in order to verify the integrity of the log file, $\mathcal{V}$ has to know the initial key to verify all entries in sequential order. If the log entry and the MAC do not correspond, the log file has been corrupted from this moment on, and any entry afterwards is no longer trustworthy.

However, the strict use of forward integrity also prohibits authorized changes to log entries; for example, if personal data shall be removed from log entries after seven days, the old MAC must be thrown away and a new MAC has to be calculated. While this is not a big issue from a computational complexity perspective, it means that the integrity of old log entries may be violated during this rollover if $\mathcal{U}$ has meanwhile been compromised.

C. Encrypted log files

Schneier and Kelsey [5] developed a cryptographic scheme to secure encrypted log files. They motivated the approach for encrypting each log entry with the need of confidential logging, e. g., in financial applications. Figure 3 shows the

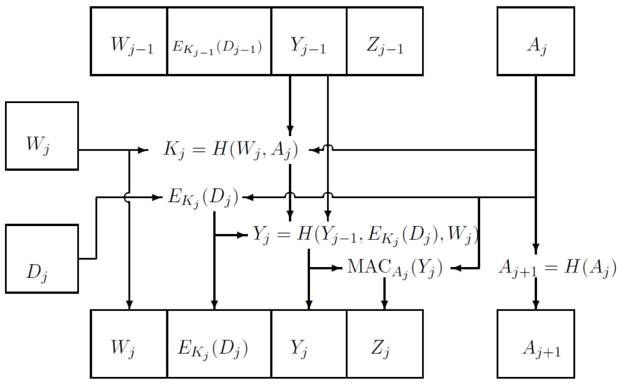


Fig. 3. The Schneier and Kelsey approach taken from [5]

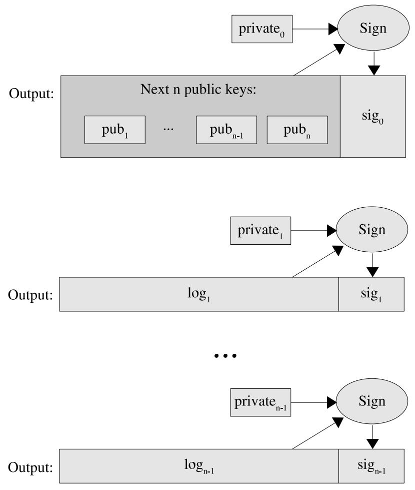


Fig. 4. The Holt approach taken from [6]

process to save a new log entry. Any log entry D_j on $\mathcal{U}$ is encrypted with the key K_j, which in turn is built from the secret A_j (in the paper s_j) and an entry type W_j. This entry type allows $\mathcal{V}$ to only verify predefined log entries. There is also some more information stored in a log entry, namely Y_j and Z_j, which are used to allow the verification of a log entry without the need of decryption of D_j. Therefore, only $\mathcal{T}$ is able to modify the log files.

However, this approach does not allow for the deletion method of log entries or parts thereof because then the verification would inevitably break.

D. Public key encryption

Holt [6] used a public/private-key-based verification process in his approach to allow a complete disjunction of $\mathcal{T}$ and $\mathcal{V}$. Therefore, a limited amount of public/private-key pairs are generated. The public keys are stored in a meta log entry, which is signed with the first public key, which should be erased afterwards securely. All other log entries are also signed with the precomputed private keys. If there are no more keys left, a new limited amount of public/private-key pairs are generated.

The main benefit of this approach is that the verifier $\mathcal{V}$ can not modify any log entry because he only knows the public keys, which can be used for verifying the signature but does not allow any inference on the used private key.

E. Aggregated Signatures

In scenarios where disk space is the limiting factor it is necessary that the signature, which protect each single log entry do not take much space. In all of the approaches sketched above, the disk usage by signatures is within $O(n)$, where n is the total amount of log entries. To deal with a more space-constrained scenario, Ma and Tsudik [7] presented a new signature scheme, which aggregates all signatures of the log entries. This approach uses archiving so that the necessary disk space amount is reduced to only $O(1)$.

The main drawback of this approach is that a manipulation of a single log entry would break the verification process but the verifier is not able to determine, which (presumably modified) entry causes the verification process to fail. As a consequence, it is also not possible to delete or to modify log entries, e.g., to remove personal data after reaching the maximum retention time.

IV. The SLOPPI framework

The survey of the related work shows that there is no solution yet that fulfills both necessary characteristics for log files: *integrity* and *compliance*. This approach combines key operations from those previous approaches in a new innovative way to achieve both characteristics. As introduced in Section II, a couple of types of log files, which are all handled a bit differently, are used for the framework.

First, the application log file L_a can be protected by any approach presented in Section III. After, for example, seven days these log files can also be deleted for compliance reasons. It is also possible to use log rotation techniques to fulfill local data protection policies. It is necessary to mention that any information about an attacker, which is not detected during this period, will be lost and cannot be recovered. But this is not a drawback of the presented framework because this is necessary to fulfill the data protection legislation especially in Europe or in Germany, which mandates to erase any privacy protected data after seven days. In scenarios where the log files can only be read after a longer offline period, e. g., low power sensor-networks devices, the period to delete log files should be set individually so an administrator is able to analyze any log data before they are deleted.

In the next step, the master log file L_m has to be secured. As it has only a few entries a day, it can be protected a public key scheme, e. g., RSA, to protect the log entries, which is described in detail in the upcoming Section IV-A. In the following work, the two keys of a public key scheme are called *signing key* (k_{sign}) and *authentication key* (k_{auth}).

Last but not least the daily log file L_d is considered in Section IV-B. Similar to the master log file, it only has very few entries per day, but they already must be considered too many entries for using public key schemes, so a symmetric key scheme is mostly the best choice.

A. Master Log

To protect L_m, the following steps are followed:

- **Log initialization.** Whenever a new master log is initialized, $\mathcal{U}$ generates a authentication key (k^1_{auth}) and a signing key (k^1_{sign}), and sends k^1_{auth} to $\mathcal{T}$ over a secure connection, e. g., a TLS connection. k^1_{auth} and k^1_{sign} are the actual used secret. $\mathcal{U}$ can now initialize the log file by saving the first message `STARTING LOG FILE` in the log file as described next.
- **Saving new log entries.** Let m be the log message of the log entry to be stored in the log file. Between saving the last entry and the actual one, it can be assumed that there is enough time to generate a new authentication/signing key pair ($k^{n+1}_{\text{auth}}, k^{n+1}_{\text{sign}}$), while ($k^n_{\text{auth}}, k^n_{\text{sign}}$) is the actual secret. $\mathcal{U}$ now generates the log entry $m^* = (timestamp, m, k^{n+1}_{\text{auth}})$ and computes $\text{Enc}_{k^n_{\text{sign}}}(m^*)$. The last result is the new log entry, which is written to the log file. Immediately after calculating the encrypted result, the key pair (k^n_{sign},k^n_{auth}) is erased securely. Only the encrypted parts of the calculation are written to the log file.
- **Closing the log file.** If the master log file has to be closed, the last message `CLOSING LOG FILE` is saved into the log file. It is important that in this case it is not necessary to generate a new key pair and to save the next authentication key into the log file.

As L_m is used as meta log, which does not contain any application or system messages, we use message m. As mentioned before, the daily log is encrypted with a symmetric crypto scheme. Every day a new daily log is initialized by the system. The name and location of the created daily log is p_1. Furthermore p_2 is the first entry in L_d and finally p_3 appoints the necessary key for the log initialization step. m is then the concatenation of p_1, p_2, and p_3, e. g., `/var/log/2012-12-21.log;STARTING LOG FILE;VerySecretKey` together with $H(p_1, p_2, p_3)$.

Because of the need to detect manipulations of the L_m, it is necessary that m also contains a hash value of p_1, p_2, and p_3. With the knowledge of $H(p_1, p_2, p_3)$ it is possible to detect, where the decryption process failed.

To verify an existing master log, it is necessary to use the authentication key saved during the generation of the log file. With this key it is possible to decrypt the first entry, which leads to the next authentication key. This step can be performed until the actual last message or the `CLOSING LOG FILE` entry is reached. Because m consists of the necessary information about the daily log files, it is possible to verify any daily log that is still available. If a daily log has already been deleted, then this daily log and the connected application logs cannot be verified any more, but it is still possible to use the upcoming log entries of L_m.

B. Daily Log

Similar to the master log, the daily log is also processed in the three steps:

- **Log initialization.** Every day a new daily log has to be initialized. The above described systems $\mathcal{U}$ and $\mathcal{T}$ are in this case $\mathcal{U} = L_d$ and $\mathcal{T} = L_m$. $\mathcal{U}$ generates a symmetric key k_{sym} and sends k_{sym} to $\mathcal{T}$ together with the name and path of the actual log file and also the first message `STARTING LOG FILE`. The actual used secret is now k_{sym}. $\mathcal{U}$ can then initialize the log file by saving the first message `STARTING LOG FILE` in the log file as described next.
- **Saving new log entries.** As above, a symmetric key scheme is used for the daily log, e. g., `AES`. Let m be the message that has to be stored in the log entry and k_{sym} the actual used secret key. $\mathcal{U}$ now randomly choses a new secret key $k^{\text{new}}_{\text{sym}}$. Because of the use of symmetric key schemes, this step is not computationally expensive. The new log entry now is $\text{Enc}_{k_{\text{sym}}}(m^*)$ with the content $m^* = (timestamp, m, k^{\text{new}}_{\text{sym}}, H(timestamp, m, k^{\text{new}}_{\text{sym}}))$, which is written to the log file. The hash value is stored for verification purpose, so it is possible to detect the exact log entry, where a manipulation took place.
- **Closing the log file.** If the master log file has to be closed, the last message `CLOSING LOG FILE` is saved. This message, the file name and location, the MAC of the entire log file, and the last generated key are sent to the master log.

In the daily log, there are only three types of messages besides `STARTING LOG FILE` and `CLOSING LOG FILE`:

- `START APPLICATION LOG`. It contains a timestamp (in plain text), the file name and location of the log (plain text), the initialization key of the application log (encrypted), the first message of the application log (encrypted), and the file name and location of the log (encrypted). The encrypted parts of the message are condensed in one encryption step.
- `STOP APPLICATION LOG`. Similar to the start message, this message contains a timestamp (plain text), the file name and location of the log (plain text), the last key of the application log (encrypted), the last message of the application log (encrypted), and the file name and location of the log (encrypted). The encrypted parts are again condensed in one encryption step.
- `ROTATING APPLICATION LOG`. In case of a log rotation procedure this message contains a timestamp (plain text) and both file name before and after a rotation (plain text). As in the previous message type there are also the file names and locations in ciphertext.

It is important that all application logs, which have their starting message in a daily log, have to write their stopping message also to the same daily log. This is the reason why some contents in the log file are still in plain text. Otherwise the logging engine would have to remember, which application log is connected to which daily log. This also means that it is possible that a daily log is still open when the next daily log is initialized. The `ROTATING APPLICATION LOG` message could be in later daily logs because the specifics of the log rotation algorithm are not known and it could be that a log rotation is performed only once a week.

The main reason to use the daily log is to reduce the space requirements of the main log. It is quite unusual that the main log is initialized for a second time if the system is running normally. There are round about two entries per day, which have to be stored over a long time. The daily log could be deleted after all application logs mentioned in this specific daily log are deleted. Depending on the amount of running applications on a server it is not unusual that there is more than one application log used on a system.

V. Verifying log entries and security analysis

To verify log entries, the initial master key is needed. As described above, each log entry in the master log is encrypted as $\mathrm{Enc}_{k^n_{\mathrm{sign}}}(m^*)$ with $m^* = (timestamp, m, k^{n+1}_{\mathrm{auth}})$. To decrypt the message only the authentication key is needed, which is stored at the log file initialization step. After the first log entry is decrypted, the authentication key to decrypt the second log entry is obtained and so on. The first time the next log entry could not be decrypted shows a manipulation of the log files, which causes by an attacker or a malicious administrator who has tried to blur his traces.

This verification step is to verify the master log and to obtain the verification keys for the daily log. As the entries in the daily log looks like $\mathrm{Enc}_{k_{\mathrm{sym}}}(m^*)$ with the content $m^* = (timestamp, m, k^{\mathrm{new}}_{\mathrm{sym}})$ the first entry could be decrypted by using the symmetric key stored in the master log. The symmetric key for any other entries are in the message payload of the previous log entry. As above in the master log, it is not possible for an attacker to modify any log entry in such a way that the encryption step works correctly.

To verify the application log it is necessary to take a look at Section III as it depends on the method used to protect the application logs. Generally it is necessary to use the secret stored in the daily log. In the following a short security analysis on the approach is shown. This analysis does not regard the application log as the security analysis in the original papers are sufficient.

On the one hand it is not possible for an attacker to gain information from the daily log or the master log as they are both encrypted with well known crypto schemes. On the other hand it is not possible to delete any entry of the log files because during the verification step, the decryption of the entry would fail and the manipulation would become evident. Furthermore, assumed that there is no implementation bug, any used key is deleted immediately so no attacker could restore it.

The only possibility of an attacker is to be fast enough to gain access to the system and to shut the logging mechanism out of service before any log entry is written to the disk. This could happen when the attacker is trying a DoS attack on the system before he is breaking in. Standard implementations of the logging service of the system are able to prevent the system from this method of attack.

VI. Conclusion

Data protection and privacy laws in Europe and several court verdicts in Germany demand the deletion of personal data from log files after a given retention time, which is a use case that is not supported by previous secure logfile management approaches. Motivated by these legal issues and log management requirements in the pan-European SASER-SIEGFRIED project, SLOPPI is a novel cryptographic logging framework that supports the deletion of old log entries without the need to re-calculate message authentication codes and to re-write the remaining log file entries. After outlining the requirements for a secure logging solution, establishing a terminology, and reviewing related work, the SLOPPI framework was presented in this paper along with its primary components, the master log, the daily logs, and the application of cryptographic functions to make the resulting logging solution tamper-evident. Finally, the verification scheme and relevant attack paths were discussed.

In the next step, SLOPPI will be implemented in the SASER project to gain first practical experiences in both setup variante, i. e., with and without a reliable central log server. SLOPPI will also be extended to facilitate the partial deletion of log entries, i. e., only those parts of log entries that contain personal data will be removed after reaching the retention period, whereas the rest of each log entry can be retained for an arbitrary longer time. Besides the removal of personal data, SLOPPI will also be extended to support anonymization and pseudonymization of personal data. The implementation will be made available as open source.

Acknowledgment

Parts of this work has been funded by the German Ministry of Education and Research (FKZ: 16BP12309).

References

[1] W. Hommel, S. Metzger, H. Reiser, and F. von Eye, “Log file management compliance and insider threat detection at higher education institutions,” in *Proceedings of the EUNIS’12 congress*, Oct. 2012, pp. 33–42.

[2] The SASER-SIEGFRIED Project Website. [retrieved: 03.04.13]. [Online]. Available: http://www.celtic-initiative.org/Projects/Celtic-Plus-Projects/2011/SASER/SASER-b-Siegfried/saser-b-default.asp

[3] S. Metzger, W. Hommel, and H. Reiser, “Migration gewachsener Umgebungen auf ein zentrales, datenschutzorientiertes Log-Management-System,” in *Informatik 2011*. Springer, 2011, pp. 1–6, [retrieved: 03.04.13]. [Online]. Available: http://www.user.tu-berlin.de/komm/CD/paper/030322.pdf

[4] M. Bellare and B. S. Yee, “Forward integrity for secure audit logs,” Department of Computer Science and Engineering, University of California at San Diego, Tech. Rep., Nov. 1997.

[5] B. Schneier and J. Kelsey, “Cryptographic support for secure logs on untrusted machines,” in *Proceedings of the 7th conference on USENIX Security Symposium*, vol. 7. Berkeley, CA, USA: USENIX Association, Jan. 1998, pp. 53–62.

[6] J. E. Holt, “Logcrypt: forward security and public verification for secure audit logs,” in *ACSW Frontiers*, ser. CRPIT, R. Buyya, T. Ma, R. Safavi-Naini, C. Steketee, and W. Susilo, Eds., vol. 54. Australian Computer Society, Jan. 2006.

[7] D. Ma and G. Tsudik, “A new approach to secure logging,” in *ACM Transactions on Storage*, vol. 5, no. 1. New York, NY, USA: ACM, Mar. 2009, pp. 2:1–2:21.

TeStID: A High Performance Temporal Intrusion Detection System

Abdulbasit Ahmed, Alexei Lisitsa, and Clare Dixon
Department of Computer Science
University of Liverpool
Liverpool, United Kingdom
{Aahmad, Lisitsa, CLDixon}@liverpool.ac.uk

***Abstract*—Network intrusion detection systems are faced with the challenge of keeping pace with the increasingly high volume network environments. Also, the increase in the number of attacks and their complexities increase the processing and the other resources required to run intrusion detection systems. In this paper, a novel intrusion detection system is developed (TeStID). *TeStID* combines the use of high-level temporal logic based language for specification of attacks and stream data processing for actual detection. The experimental results show that this combination efficiently make use of the existing testing machine resources to successfully achieve higher coverage rate in intensive network traffic compared with *Snort* and *Bro*. Additionally, the solution provides a concise and unambiguous way to formally represent attack signatures and it is extensible and scalable.**

***Keywords*-network intrusion detection system; temporal logic; parallel stream processing; runtime verification**

I. Introduction

Intrusion Detection Systems (IDS) detect intruders' actions that threaten the confidentiality, availability, and integrity of resources [1]. Network Intrusion Detection Systems (NIDS) reside on the network, and are designed to monitor network traffic. An *NIDS* examines the traffic packet by packet in real time, or close to real time, to attempt to detect intrusion patterns [2].

The increasing network throughput challenges the current *NIDS* running on customary hardware to monitor the network traffic without dropping packets. Consequently, many attacks are not detected by the current *NIDS* [3], [4]. Some vendors provide a highly specialized and configured hardware to prevent the drop of packets [5], [6].

The techniques developed to solve the problem of IDS reliability due to packets loss in high-speed networks can be grouped into the following:

- Data reduction techniques (i.e., data based approach) [7].
- Load balancing, splitting, or parallel processing of traffic (i.e., distributed/parallel execution based approach) [3].
- Efficient algorithms for pattern matching (i.e., algorithm based) [8].
- Hardware based approach such as using graphics processing units [9] or field-programmable gate array (FPGA) devices [10].

Some works use combinations of the above techniques as in [8] where the algorithm is hardware implementable. The research area is still active and there is no solution that can keep up with the increase in bandwidth.

Another issue with current IDS is that the attack signatures are often specified in rather low-level languages and, especially, in the case of multiple packets attacks, may become cumbersome and error-prone, difficult to write, analyse and maintain. For instance, *Bro* [11] has special scripting language for detecting multiple packet attacks.

In this paper, we present a new Temporal Stream Intrusion Detection System (TeStID), the design of which addresses both the issue of efficiency and reliability in high-speed networks and the issue of the high-level and unambiguous specifications. In *TeStID* a multiple packet attack is represented with a formula, whereas in *Bro* one page of code or more might be needed to represent the same attack. The system uses Temporal Logic (TL) [12] for the attack signature specifications and available Stream Data Processing (SDP) [13]–[15] technology for the actual detection. *TL* is the extension of classical logic with operators that deal with time which allow us to formally specify temporal events (i.e., network packets) as they traverse the network. The *SDP* is a database technology applied to streams of data which are designed with processing capabilities suitable for data intensive applications.

We use Many Sorted First Order Metric Temporal Logic (MSFOMTL), which was defined in [16] to represent data packets arriving over time and attack patterns. This allows us to state, for example, "a packet arrives between 6 and 9 seconds".

In [16], we described the architecture of *TeStID* and provided preliminary experimental results using the DARPA IDS Evaluation Data [17]. The experiments and case studies focussed on the detection of multiple packet attacks.

In this paper, we extend our work and provide further detailed experimental analysis considering both single and multiple packet attacks, allowing single packet attacks with payload and experimenting with the high performance features available in stream data processing. Additionally, we compare the performance of TeStID with two well known open source NIDSs: *Snort* [18] and *Bro* [11].

The rest of the paper is organized as follows. Section

II presents an overview of the proposed system and its design. Section III presents the experiment setup environment. Section IV presents the experimental results with and without using the high performance features of stream data processing. Related work is presented in Section V. Finally, Section VI concludes this paper and discusses future work.

II. TeStID Overview and Design

In *TeStID* attack signatures are formally represented using temporal logic. The signature based intrusion detection problem is reduced to the problem of checking whether a temporal formula ϕ representing an attack pattern s true in a temporal model M representing a linear sequence of all received (observed) network packets, that is $M \models \phi$?

In our model, we are dealing with finite initial segments of potentially infinite sequences. This and the fact that properties considered are time bounded makes the decidability of model checking trivial.

In the TeStID system the TL formulae specifying attacks are automatically translated into stream SQL (SSQL) language constructors. We use StreamBase [15] as the stream data base engine and it uses *SSQL* as the stream query language. Consequently, this *SSQL* code is executed to detect temporal patterns specified in the original formula in the incoming events.

The syntax of temporal logic *MSFOMTL* used in the system is as follows. An atomic formula has the form $P(te_1, te_2, \dots, te_n)$ where P is a predicate and for $i = 1, \dots, n, te_i$ is a term. The atomic formulae represent the information about individual packets and terms correspond to the fields within a packet.

The syntax of *MSFOMTL* formulae are defined as follows:

$$\begin{aligned}\mathcal{L} :=& P \mid \varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \Diamond_{[t_1,t_2]} \mid \Box_{[t_1,t_2]} \\ & \mid \blacklozenge_{[t_1,t_2]}\varphi \mid \blacksquare_{[t_1,t_2]}\varphi \mid (\forall x)\varphi \mid (\exists x)\varphi \qquad (1)\end{aligned}$$

In addition to the usual connectives of propositional logic these formulae include bounded temporal operators “always in the time between t_1 and t_2” $\Box_{[t_1,t_2]}$, “eventually in the future in a time between t_1 and t_2” $\Diamond_{[t_1,t_2]}$, “sometime in the past in a time between t_1 and t_2 before now” $\blacklozenge_{[t_1,t_2]}$, and “always in the past in all time between t_1 and t_2” $\blacksquare_{[t_1,t_2]}$. The incoming events form the temporal models, $\mathcal{M} = \langle \mathcal{T}, <, \mathrm{I} \rangle$ where:

- $\mathcal{T} = \{\tau_0, \tau_1, \dots\} \subset \mathbb{R}^+$, where $\mathbb{R}^+$ is a non-empty set of positive real numbers and $\mathcal{T}$ is the set of all arrival moments.
- $<$ is a linear order on $\mathcal{T}$.
- I is an interpretation which maps $\mathcal{T}$ into the set of all possible packets $[\![\mathcal{P}]\!]$: $\mathrm{I} : \mathcal{T} \to [\![\mathcal{P}]\!]$

So, I(τ_i) represents a packet arriving at a moment $\tau_i \in \mathcal{T}$.

III. The Experiment Setup

The setup of the test environment closely resembles the actual deployment of NIDS. In a typical NIDS deployment, the network sensor device receives a copy of all the traffic that traverse the network. The testing environment is setup as follows:

- *TeStID* is installed on an INTEL® Core$^{\text{TM}}$ i5 2.26 GHz machine with 4 GB of memory and a Gigabit Network interface that is capable of running in promiscuous mode (i.e., listening to all network traffics). *TeStID* was developed with *StreamBase* developer version 7.1.
- Another computer (INTEL® Core$^{\text{TM}}$2 Quad Processor Q6600 and 2 GB memory) with a Gigabit Network interface is used to replay the data.
- TCPREPLAY [19] was used to replay the trace files. TCPREPLAY is a tool that replays TCP dump files [20] at specified speeds onto the network.
- A switch to connect the two PCs or simply crossover network cable.
- A custom data file, which was prepared using a DARPA dump test file and some test files from the free license version of Traffic IQ Professional$^{\text{TM}}$ [21].
- *Snort* version 2.9.1, and *Bro* version 1.5.1.

The data file has 3,014,600 packets. It has 50 different single packet with payload attacks which are distributed over 792 packets. This means 792 total instances of the 50 attacks.

During the experiments, the send and receive buffers were increased to ensure that all packets are sent successfully and are received with no loss of packets. The reading buffer parameters rmem_max and rmem_default increased significantly to 110 MB. Also, the writing buffer buffer parameters wmem_max and wmem_default increased to the same value. Using these values, TCPREPLAY successfully replayed the dump files at top speed multiple, that is, the maximum that can be send on the designated hardware. On the receiving end, TCPDUMP successfully captured all the packets. This was important tuning step as *Snort*, *Bro*, and *TeStID* use TCPDUMP to sniff packets.

IV. The Experimental Work

The experiments were run on *Snort*, *Bro*, and *TeStID* using the test data file. These attacks were coded in *Snort* and *Bro* according to syntax specified in their documentations [11], [18]. For *TeStID* these attacks are written in a file using *MSFOMTL* syntax. Then the translator is run to translate these formulae into *SSQL* code. A typical example, is *Snort* attack id 255 “DNS Zone Transfer TCP”. This attack is identified when the destination port (x_4) is 53 and the payload (x_{12}) contains a string that is identified by the regular expression `".{14}.*\u0000\u0000\u00fc.*"` which means the string must contains any 14 characters, possibly followed by an additional character, followed by two null characters, and “ü”. Formally, it is represented in *MSFOMTL* as follows:

$$(\exists x_4, x_{12})((\exists y_1, y_2, y_3, y_5, y_6, y_7, y_8, y_9, y_{10}, y_{11}) \\ P(y_1, y_2, y_3, x_4, y_5, y_6, y_7, y_8, y_9, y_{10}, y_{11}, x_{12}) \wedge (x_4 = 53) \\ \wedge (x_{12} = f(\texttt{".\{14\}.*\textbackslash u0000\textbackslash u0000\textbackslash u00fc.*"}))) \tag{2}$$

The meaning of the terms y_1,y_2,y_3,y_5,y_6,y_7,y_8,y_9,y_{10},y_{11} are source IP address, source port, destination IP address, sequence number, acknowledgement number, ack flags, syn flag, reset flag, push, and urgent flag, respectively. These are free terms and can take any valid value from its corresponding sort domain.

When translating the above formula into *SSQL* we obtain the following code:

```
CREATE STREAM out_Input;
APPLY JAVA "TCP_W_Payload" AS Input (
 schema0 = "<?xml version=\"1.0\" encoding=
 \"UTF-8\"?>\n<schema name=\"schema:Input\">
\n
<field description=\"\" name=\"x1\"
type=\"string\"/>\n
.
.
<field description=\"\" name=\"x12\"
type=\"string\"
/>\n</schema>\n"
)
INTO out_Input;

CREATE OUTPUT STREAM Output;
SELECT * FROM out__Input
WHERE (out_Input.x4 = 53)
and (regexmatch(".{14}.*\u0000\u0000\u00fc.*"
     , x12))
INTO Output;
```

Each of the 50 attacks produce similar code to the above. There are two main parts in each code: the input adapter call (APPLY JAVA) and the select statement (from here onward we refer to it as the filter code).

A. The Experiments Results: multiple packet attacks

The results of experiments on multiple packet attacks using *TeStID* were presented previously in [16]. The results obtained is reproduced in Table I. The first column contains the attack names, the second column contains the DARPA [22] data files used, the third column contains the number of attacks detected in normal replay, the fourth column contains the number of attacks detected in 1350 speed multiple of normal recorded speed (the highest speed achieved without packet loss), and the last column is the actual number of attacks in the data. The bandwidth achieved is 524 Mbits/Sec and the peak number of packets was about 250,000 packets per second. In *Snort* the specification of multiple packet attacks is not possible but in *Bro* it is feasible. These attacks need to be written in *Bro* scripting language. We ran the script for the Syn Flood or Neptune which comes with *Bro*

Table I. Results of Multiple Packet Attacks

Attack Name	Data Set	Packets Replayed	Normal Replay	1350X Normal	Actual No. of Attacks
DoSNuke	01/04/99 in	2,356,503	1	1	1
	05/04/99 in	2,291,319	2	2	2
	06/04/99 in	3,404,824	1	1	1
Neptune	05/04/99 in	2,291,319	1	1	1
	06/04/99 out	2,558,481	3	3	3
	09/04/99 in	3,393,918	1	1	1
TCPReset	06/04/99 in	3,404,824	2	2	2
	07/04/99 in	2,087,942	1	1	1
	09/04/99 in	3,393,918	1	1	1
ResetScan	08/04/99 in	3,201,381	2	2	2

default installation. *Bro* achieves about 300 Mbits/Sec and about 76,000 packets/Sec using the same test data file in Table I and the same testing environment. *Bro* crashed within seconds at the speed multiple of 1350 after consuming the available resources on the testing machine.

In this paper, the experiments on *TeStID* are further extended to explore its capabilities to detect single packet attacks with payload. The experiments on these attacks can be grouped into two sets. In the first set, no high performance features are used. In the second set the high performance features are used. The following sections give more detail and the results of each set.

B. The Experiments Results: single packet attacks

Here, we present the results of the experiments with the detection of single packet attacks with payload. In this type of attacks the packets are inspected deeply and not only the headers as in the multiple packet attacks. Three different variants of the execution of *SSQL* code were considered.

In *StreamBase* a basic execution unit running on the server is called a container. The *SSQL* codes run inside this container which has a name and a set of associated handling processes that are created by the stream server. In this set of experiments there are three possible implementations to run single packet attack detection on the *StreamBase* server without using the high performance features.

- In the first variant of implementation, each translated specification of an attack is written as an independent program that runs in its own container.
- In the second implementation each translated specification of an attack is an independent program but all programs are using the same container.
- In the third implementation all the translated specifications of the attacks are written as one program and run in one container sharing the input adapter.

The results obtained for this set of experiments are shown in Table II. Each experiment is an average of three runs and before each run the machine was restarted. The first column shows the tested implementation. Columns two through four contain the results of running each system while using multiple replay speeds of 2, 4, and 8 respectively, to send the packets (blank if no test is done).

Table II. results without using high performance features

Implementation	X 2	X 4	X 8
50 independent programs files	-6	-139	-384
50 programs in one file	0	-116	-323
50 programs sharing input adapter	0	-6	-74
BRO v1.5.1	0	0	-1
SNORT v2.9.1	-119	-125	

In the first row, 50 different program files are run where each program corresponds to one attack. This implementation gives the worst coverage rate as it misses six attacks in the speed multiple of two (x 2 column). The result makes sense as each program needs to capture the packets with the input adapter code which in turns calls a JAVA code that interfaces with the network interface and then the rest of the program processes the packets (tuples) in sequential fashion. This scenario is the same for all the 50 programs which means each program will have its own space (parallel region or container in *StreamBase* terminology). The *StreamBase* engine suffers from the overhead of having to interact with all these regions.

The second row shows the results of running the code for all 50 attacks in one container (i.e., the attacks are all written in one file). This implementation gives slightly better results as it misses 116 attacks in the speed multiple of four. With this implementation the *StreamBase* engine achieves more efficient processing with the decrease of the inter process handling overhead.

In the third row, all the 50 filter codes are in one file but only one input adapter code is used. All the filtering codes read from the same input adapter. This means less resources from the system are used. In *StreamBase* when a tuple (packet) arrives it must be processed till completion before the next tuple arrives. This means the input adapter feeds the tuple to the first coded filter and then coded map of the first attack. Then the codes of the second and so on. If another tuple (packet) arrived, it will be retained in a buffer until the first processing is finished. This type of implementation was the best without the use of concurrency and multiplicity options as it misses only 6 attacks at the multiple speed of four. Unfortunately, this is worse than what *Bro* achieved in the fourth column. *Bro* misses only one attack in the speed multiple of eight. The *Snort* result is shown in the fifth column and it misses 119 attacks in the speed multiple of two. It was worse in speed multiple of four as it missed 125. The blank in the table for speed multiple of eight for *Snort* means that test was not done as we thought it is not necessary.

These results gave the motivation to investigate the high performance features of *StreamBase* and thus a second set of experiments were carried out as described in the next section.

V. Investigating High Performance Features

In general, stream data processing engines have high performance features. *StreamBase* has concurrency and multiplicity options that can be used to allow us to achieve higher performance. In this set of experiments we use these features. First of all we explain the following related definitions to the high performance features used:

- Concurrency means part of the code runs in its own thread.
- Multiplicity refer to the number of instances of the code.
- Dispatch style is related to the multiplicity. The dispatch style specifies how each instance receives data tuples: in round robin, broadcast, or based on a data value. In broadcast each instance will receive a copy of the incoming tuple. In round robin, the first tuple goes to the first instance and the second goes to the second and so on. Based on value is by checking the value against a test condition and then dispatching to the designated instance for that value.

In *StreamBase* the concurrency, the multiplicity, or both can be set. According to *StreamBase* manual, these options can be used for portions of the application if the code portion is long-running or compute-intensive, can run without data dependencies on the rest of the application, and it would not cause the containing module to be waiting or blocked.

In this set of experiment, each *SSQL* code for detecting an attack in the previous section contains two components: the input adapter and filter code. For the input code, we can use the concurrency option but not the multiplicity as it has no input stream. For the filter code part we can use both options. This means that there are eight possible implementations as can be seen in Figure 1. Table III shows

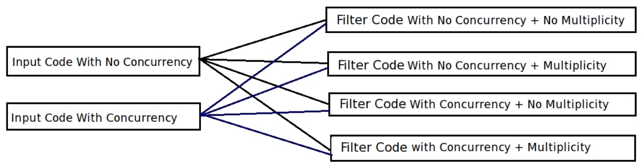

Input Code With No Concurrency
Input Code With Concurrency
Filter Code With No Concurrency + No Multiplicity
Filter Code With No Concurrency + Multiplicity
Filter Code With Concurrency + No Multiplicity
Filter Code with Concurrency + Multiplicity

Figure 1. All Possible Implementations Using Concurrency/Multiplicity

the results of the experiments of this category. Notice that the experiments start at speed multiple of 8 as we try to achieve better result compare to *Bro* in Table II. Table III (CC) denotes "Concurrency". (NC) denotes "No Concurrency", (1) denotes single instance or no multiplicity, and (n) denotes n multiplicity where n is an integer such that $n > 1$. For instance, the first row shows the results of running non concurrent input code (NC) and non concurrent (NC) 50 filter codes of (1) instance each (i.e., no multiplicity). The following are observations on these results:

Table III. Experiments Results Using Concurrency and Multiplicity

	Implementation	X8	X16	X24	X48
1	NC Input Code + 50 NC Filter Code (1)	-79			
2	NC Input Code + 50 NC Filter Code (2)	-67			
3	NC Input Code + 50 CC Filter Code (1)	-293			
4	NC Input Code + 50 CC Filter Code (2)	-299			
5	CC Input Code + 50 CC Filter Code (1)	-126			
6	CC Input Code + 50 CC Filter Code (2)	-128			
7	CC Input Code + 50 NC Filter Code (1)	0	0	0	-5
8	CC Input Code + 50 NC Filter Code (2)	0	0	0	-2
9	CC Input Code + 50 NC Filter Code (3)	0	0	0	-1
10	CC Input Code + 50 NC Filter Code (5)	0	0	0	-2
11	CC Input Code + 50 NC Filter Code (10)	0	0	0	-5

- The first row result is almost the same result obtained previously with no concurrency and no multiplicity (the third row in Table II). The difference is that we implement the attack code as a module and we did not use the concurrency or the multiplicity.
- Using 50 concurrency for the filter code give the worst result (row 3-6).
- Using input code with concurrency and no concurrency for the filter code gives better results in general (rows 7-11).
- The reason for the bad performance results when using 50 CC (row 3-6) compared with using 50 NC for the filter code (row 7-11) is that the system running in the CC implementation maintains many thread-switches per attack vs. no switch per attack with NC implementations.
- Row number 9 has the result of best performance where three non concurrent instances of filter codes are used. This is consistent with *StreamBase* rule of thumb that is for best performance the number of instances should be equal to the number of cores on the machine or less. So, we have three instances in addition to the input code running on four cores on the testing machine.
- In rows 7-8 less than three number of instances used and in rows 10-11 more than three instances used. Increasing or decreasing the number of instances from three cause the performance to degrade.

This set of experiments using the concurrency and multiplicity features of *StreamBase* enable us to achieve a result that exceeded the results of *Snort* and *Bro* which are presented in Table II. Furthermore, the testing machine has four cores, but the solution can take advantage of using more cores. *Snort* doe

VI. Related Work

Temporal logic is used in the network based IDS *MONID* [23] and *ORCHIDS* [24]. *MONID* a prototype tool based on *Eagle* [25]. *Eagle* is a runtime verification or runtime monitoring system that uses finite traces. Simply, it monitors the execution of a program and checks its conformity with a requirements specification, often written in a temporal logic or as a state machine. Naldurg, Sen, and Thati [23] propose the use of *Eagle* in online intrusion detection systems. In *MONID*, *TL* is used to represent a safety formula ϕ (specification of the absence of an attack) and the system continuously evaluates ϕ against a model M representing a finite sequence of events. Whenever ϕ is violated (i.e., $M \not\models \phi$) an intrusion alarm is raised.

ORCHIDS [24] is a misuse intrusion detection tool, capable of analyzing and correlating temporal events in real time. *ORCHIDS* uses an online model checking approach. *ORCHIDS* uses temporal logic to define attacks that are complex, correlated sequences of events, which are usually individually benign. The attack signatures are represented or described in the system as automata. The *ORCHIDS* online algorithm matches these formulae against the logs and returns enumerated matches [24]. Similar to *MONID*, *ORCHIDS* reads events from many sources that have different formats (networks and system logs) and this makes representing attack signatures difficult in standard methods. As far as we know from published paper [24], the tests for *ORCHIDS*, mainly concentrated on the proof of concept and not performance.

The main differences between the system we described in this paper as compared with *MONID* and *ORCHIDS* is that the model checking problem ($M \models \phi$) is reduced to the stream query evaluation, which is subsequently executed by high-performance *SDP* engine. Also, in the proposed system, the way of using temporal logic takes advantage of its expressiveness and conciseness to allow the user to express attack signatures transparently and independently from the underlying technical implementations. We could not test *MONID* and *ORCHIDS* in the same testing environment we used in our experiments, as their implementations were not available.

Snort [18] and *Bro* [11] are the oldest and most popular open source NIDSs known today. According to the *Snort* organization site, over 4 million downloads and more than

400,000 registered users show the wide popularity of *Snort* as a deployed IDS solution. Specifying multiple packet attacks is not possible in *Snort*, but it is possible in *Bro*. *Bro* is considered highly stateful (i.e., it keeps track of each established session states) and was developed primarily as a research platform for intrusion detection and traffic analysis. It has no subscription service where users can download new attack signatures like what the *Snort* community provides.

Data stream processing was suggested to be used in intrusion detection by the *StreamBase* Event Processing Platform™ software development team. An example of using *SB* in intrusion detection was given in the demo package in which a statistical method was used to predict anomaly behaviour of network traffic.

GIGASCOPE [26] is a proprietary data stream management system (DSMS) and was developed by AT&T and it is currently used in many AT&T network sites. *GIGASCOPE* is special purpose DSMS engine for detailed network applications. Johnson, Muthukrishnan, Spatscheck, and Srivastava [27] from AT&T research lab argued that *GIGASCOPE* can serve as the foundation of the next NIDSs because of the functionality and performance. They presented some written examples to detect Denial of Service attacks.

VII. Conclusion and Future Work

The combine use of temporal logic and stream data processing as proposed in this paper is a promising solution toward network intrusion detection system in high-volume network environments. The use of *TL* gives the system the advantage of providing a concise and unambiguous way to represent attacks. Also, using *TL* abstracts the user away from the technical requirements details. In addition, the system is extensible as it is easy to add new attacks and recompile the system to include these.

Using stream data processing gives the system the advantage of having scalable performance. As the results of our experiments suggest the system outperforms both *Snort* and *Bro* systems in high-speed network environments. The system benefits from adding additional CPUs to meet a larger volume of data. To show the feasibility and benefits of using stream processing, the *SB* development version was adequate, even though that it has less capabilities in terms of execution speed and utilizing machine resources than the server version [15].

In the future work, we are planning to extend *TeStID* to be used in *protocol anomaly based* network intrusion detection. *MSFOMTL* will be used to represent parts of the TCP protocol normal specification and any deviation from this specification will be reported at the runtime.

Acknowledgment

The first author would like to thank the Saudi Ministry of Higher Education for partially funding this research. Also, we would like to thank the Stream Base Inc. for giving us the necessary license to use their products in our work.

References

[1] W. Stallings, Network Security Essentials: Applications and Standards. Upper Saddle River, NJ: Prentice Hall, 2000.

[2] K. Scarfore and P. Mell, "Guide to intrusion detection and prevention systems (IDPS)," National Institute of Standards and Technology (NIST), Special Publication 800-94, Feb. 2007.

[3] H. Lai, S. Cai, J. Xi, and H. Li, "A parallel intrusion detection system for high-speed networks," ACNS 2004. LNCS, vol. 3089, 2004, pp. 439–451.

[4] H. Dreger, A. Feldmann, V. Paxson, and R. Sommer, "Operational experiences with high-volume network intrusion detection," in Proceedings of the SIGSAC: 11th ACM Conference on Computer and Communications Security (CSS'04), 2004, pp. 2–11.

[5] Endace Ltd., "EndaceAccess™," http://www.endace.com/, 2012, Accessed on 02 January 2013.

[6] Hewlett-Packard Development Company, L.P., "Tipping Point Digital Vaccine Services," http://h17007.www1.hp.com/docs/security/400931-004_DigitalVaccine.pdf, 2012, Accessed on 02 January 2013.

[7] K.-Y. Lam, L. Hui, and S.-L. Chung, "A data reduction method for intrusion detection," J. Syst. Softw., vol. 33, no. 1, Apr. 1996, pp. 101–108.

[8] S. Dharmapurikar, J. Lockwood, and M. Ieee, "Fast and scalable pattern matching for network intrusion detection systems," IEEE Journal on Selected Areas in Communications, vol. 24, no. 10, Oct. 2006.

[9] G. Vasiliadis, S. Antonatos, M. Polychronakis, E. P. Markatos, and S. Ioannidis, "Gnort: High performance network intrusion detection using graphics processors," in Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection, ser. RAID '08, 2008, pp. 116-134.

[10] D.-H. Kang, B.-K. Kim, J.-T. Oh, T.-Y. Nam, and J.-S. Jang, in Agent Computing and Multi-Agent Systems, ser. Lecture Notes in Computer Science, Z.-Z. Shi and R. Sadananda, Eds., 2006, vol. 4088.

[11] Lawrence Berkeley National Laboratory, "Bro Intrusion Detection System," http://www.bro-ids.org/, 2011, Accessed on 02 January 2013.

[12] M. Fisher, An Introduction to Practical Formal Methods Using Temporal Logic. Wiley, 2011. [Online]. Available: http://books.google.co.uk/books?id=zl6OLZv7d1kC

[13] A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, I. Nishizawa, J. Rosenstein, and J. Widom, "STREAM: The Stanford stream data manager," in SIGMOD Conference, 2003, p. 665.

[14] EsperTech Inc., "Esper - event stream and complex event processing for java," http://esper.codehaus.org/esper-3.3.0/doc/reference/en/html_single/index.html, 2009, Accessed on 02 January 2013.

[15] StreamBase Systems, “StreamBase Server,” http://www.streambase.com/products-StreamBaseServer.htm, 2012, Accessed on 02 January 2013.

[16] A. Ahmed, A. Lisitsa, and C. Dixon, “A misuse-based network intrusion detection system using temporal logic and stream processing,” in Proceedings of the 5th International Conference on Network and System Security, P. Samarati, S. Foresti, J. Hu, and G. Livraga, Eds. Milan, Italy: IEEE, 2011, pp. 1-8.

[17] R. Cunningham, R. Lippmann, J. Fried, S. Garfinkel, R. Kendall, S. Webster, D. Wyschogrod, and M. Zissman, “Evaluating intrusion detection systems without attacking your friends: The 1998 DARPA intrusion detection evaluation,” Defense Advanced Research Projects Agency, Department of US Defense, Technical report, 1998.

[18] Sourcefire, “SNORT,” http://www.snort.org/, 2010, Accessed on 02 January 2013.

[19] TRAC, “Welcome to TCPREPLAY,” http://tcpreplay.synfin.net/, 2010, Accessed on 02 January 2013.

[20] The TCPDUMP Group, “TCPDUMP and LIBPCAP,” http://www.tcpdump.org/, 2010, Accessed on 02 January 2013.

[21] IDAPPCOM Ltd., “Traffic IQ Library,” http://www.idappcom.com/index.php, 2012, Accessed on 02 January 2013.

[22] MIT Lincoln Laboratory, “DARPA Intrusion Detection Data Sets,” http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html, 1999, Accessed on 02 January 2013.

[23] P. Naldurg, K. Sen, and P. Thati, “A temporal logic based framework for intrusion detection,” in Proceedings of the 24th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems, Madrid, Spain, 2004.

[24] J. Olivain and J. Goubault-Larrecq, “The ORCHIDS intrusion detection tool,” in Proceedings of the 17th International Conference on Computer Aided Verification (CAV’05), 2005.

[25] H. Barringer, A. Goldberg, K. Havelund, and K. Sen, “Rule-based runtime verification,” in Proceedings of the VMCAI’04, 5th International Conference on Verification, Model Checking and Abstract Interpretation, Venice, Italy, 2004, pp. 44-57.

[26] C. Cranor, T. Johnson, and O. Spataschek, “Gigascope: A stream database for network applications,” in SIGMOD, 2003, pp. 647-651.

[27] T. Johnson, S. Muthukrishnan, O. Spatscheck, and D. Srivastava, “Streams, security and scalability,” in Proceedings of the 19th annual IFIP WG 11.3 working conference on Data and Applications Security, ser. DBSec’05, 2005, pp. 1-15.