

xMiner: Nip the Zero Day Exploits in the Bud

M Zubair Rafique and Muhammad Abulaish
Center of Excellence in Information Assurance (CoEIA)
King Saud University, Riyadh, Saudi Arabia
Email: {zrafique.c,mabulaish}@ksu.edu.sa

Abstract—Vulnerability exploits present in malformed messages are one of the major sources to remotely launch malicious activities in different protocols. Sometimes, a single malformed message could be enough to crash remote servers or to gain unfettered access over them. In this paper, we propose the design of a generic vulnerability exploits detection system **xMiner** to detect malformed messages in real time for avoiding any network hazard. The proposed **xMiner** exploits the information embedded within byte-level sequences of network messages. **xMiner** applies multi-order Markov process and principal component analysis (PCA) to extract novel discriminative features and uses them to detect attacks launched through malicious packets in real-time. The novelty of **xMiner** lies in its light-weight design which requires less processing and memory resources and makes it easily deployable on resource-constrained devices like smart phones. The system is evaluated on real-world datasets pertaining to three different protocols – HTTP, FTP and SIP. Five different classifiers are deployed to establish the effectiveness of the proposed system. On evaluation we found that the decision tree classifier performs well for HTTP and FTP datasets whereas, SVM shows highest performance in case of SIP packets.

Keywords—Network security; vulnerability exploits detection; feature extraction, machine learning.

I. INTRODUCTION

Remote attacks based on vulnerability exploitations are one of the most destructive security problems faced by the current security community. Particularly, the high profile security threat of automatic dissemination attacks or worms [1] are based on remote exploitation of vulnerabilities in compromised systems. One of the recent example of such attacks is the damaging Stuxnet worm in July 2010, which used four zero-day vulnerability exploitations to attack highly sensitive organizations (e.g. Iranian nuclear power plant) in many countries of the world [2]. Another example is the widely-spread Conficker worm in January 2009, which affected more than 15 million computers in more than 200 countries of the world [3]. Also, the successful eruption of well known CodeRed [4], Slammer [5], MSBlast [6], and Sasser [7] worms highlight the severity of damages caused in widely used services through vulnerability exploitation.

Beside the common web services, the emerging next generation networks like Voice over IP (VoIP) and IP Multimedia Subsystems (IMS) are also hot targets of remote exploitation attacks. In these services, the underlying flexibility of the application-specific protocols are exploited

to launch remote attacks [8], [9]. The devastating impact of these attacks includes successful Denial of Services (DoS) attack, crashing of hosts or end points (e.g., Smart Phones), unfettered access to the systems and remote execution of malicious codes. Recent attacks like INVITE of Death [10], Remote DoS attack on Cisco Routers [11] and Asterisks server [12], and remote code execution in video-conferencing framework of Apple Mac OS [13] highlight the severity of damages that can be caused through vulnerability exploits. Unfortunately, patching as a first line of defense is not a perfect choice since it usually requires a service disruption (reboot of a system) and causes reduction in availability¹ which severely undermines the reliability of sophisticated real-time systems [1]. Additionally, it can not be guaranteed that the applied patch will not alter the behavior of the system. As an outcome, critical services remain susceptible to novel (zero-day) exploits for a considerable period of time which cannot be effectively remunerated by applying certain administrative or technical measures.

Several Intrusion Detection Systems (IDSs) like Snort [15] and Bro [16] and fire-walls like Hogwash [17] and Shield [1] are based on misuse detection where exploits are detected by modeling the signature of *known attacks*. The main drawback of signature-based techniques is that they cannot cope up with exponential increase in new malicious exploits. Not only the size of signatures database will not scale but the time to match signatures also significantly increases [18], [19]. Last but not least, generation of signatures for one particular protocol or application can not be used in detection of zero-day exploits for wide range of services. Therefore, we argue that the design of light-weight and non-signature based detection system to identify zero-day attacks in a real-time is one of the significant research problem.

In this paper, we propose the design of a light-weight intrusion detection system, **xMiner**, which applies supervised-learning approach to detect malicious messages. The proposed system is efficient and capable to detect vulnerability exploits targeting towards next generation services at the granularity of application layer. One of the motivating factors to target application-specific services is that they are

¹A reliable VoIP infrastructure must guarantee at least 99.9% uptime to stay competitive in the telecommunication market. VoIP servers are among the SANS top 20 Security Risks [14].

widely used², constantly emerging [21] and more vulnerable to zero-day attacks [22]. Instead of using syntax-level features to model classification system, xMiner extracts novel byte-level features using multi-order Markov process and uses them to identify malicious messages. Principal Component Analysis (PCA) is applied to reduce the dimension of feature space and eliminate less discriminative features. Following are some of the salient features of xMiner:

- *Generality*: Since xMiner uses byte-level features instead of syntax-level features, it can be applied on wide-range of protocols to identify malicious messages.
- *Efficiency and real-time deployability*: The efficiency of xMiner lies in its reduced set of features that are obtained using multi-order Markov process and principal component analysis. In addition, due to its design as a *non-signature based* IDS, for predicting the class (benign or malicious) of an incoming message, xMiner is not subjected to search through a large list of signatures. Rather, it uses the trained model to predict the class of an incoming message efficiently and consequently, xMiner is real-time deployable. On experimentation, we found that for some protocols, xMiner achieves detection accuracy of more than 99% with false alarm rate of less than 0.1% in real-time.
- *Novel attack detection*: Since, rather than using signature matching approach, xMiner models a classification system to characterize the byte-level information of malicious and benign messages and uses the same learned model to detect novel (zero-day) attacks targeted towards the next generation services and devices.
- *Modularity*: The modular design approach of xMiner allows its simple yet effective deploying functionality. Due to this feature, xMiner is easily configurable on different types of application servers and devices.

We have evaluated the xMiner on three different real-world datasets related to three different services HTTP, FTP and SIP. We have deployed a real-world testbed to generate various vulnerability exploits using different types of security testing tools and scripts. The attack vectors include real-world buffer over flows, remote code execution, remote DoS and fuzzed message exploits. On evaluation, we found that in some cases xMiner achieves more than 99% detection rate and less than 0.1% false alarm rate for distinguishing benign messages from malicious messages.

The rest of the paper is organized as follows. Section II presents a summarized view of the related works on malicious message detection. The functioning detail of xMiner and its modular architecture is presented in Section III. The experimental setup, dataset description and evaluation results are presented in Section IV. Finally, we conclude the paper with an outlook to our future works in Section V.

²A market survey indicates that VoIP accounts for 49.7% of total voice traffic at the end of year 2007 [20].

II. RELATED WORK

In this section, we present a brief review of the existing network intrusion detection techniques proposed by different researchers. The protection against application-level attacks using protocol syntax was first purposed in signature-based intrusion detection systems. In [16], Paxson has proposed a signature-based IDS, Bro, which uses different protocol parsers to identify malicious packets. The developed parsers are tightly coupled with the Bro's signature engine and can not be used for wide range of services. Similarly, Roesch has proposed Snort in [15] which is also a signature-based intrusion detection system. Snort can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks targeting towards application servers. Some other signature-based intrusion detection techniques have been used by Niccolini et al. [23] and Apte et al. [24] specifically for voice over IP service based on SIP protocol. Geneiatakis et al. [25] have also proposed signature generating methodology that prevent fuzzed message attacks on VoIP infrastructure. The other signature-based techniques like binpac [26] and GAPAL [27] provide effective and generic procedures for detecting malicious packets by using protocol parsers. However, most of these techniques are dependant on signature database and hence cannot be used to detect novel attacks targeted towards wide range of services.

Düssel et al. [28] have proposed a different approach which focuses on analyzing the payloads of application-level protocols for anomaly detection. The approach detects anomalous packets by computing similarity between the attributed n-grams/tokens derived from the protocol grammar. Ingham et al. [29] have proposed a learning system on specific presentation by extracting token-based feature through delimiters specific to HTTP requests. A related work by Rieck et al. [30] proposes the design of a self-learning system for detecting malformed messages in SIP. The self-learning model operates on the tokens and n-gram based features, and the learned model from higher values of n-gram features performs better than the token-based attribute model. Kruegel et al. [31] have developed an IDS for HTTP, which uses a number of features like length, character distribution, etc. to detect malicious packets.

In contrast to the above-mentioned intrusion detection systems, xMiner uses byte-level transitions of network messages and applies multi-order Markov process to extract discriminative features. Further, it applies PCA on the extracted set of features to filter-out irrelevant features and reduce the dimension of the feature space. This boost the efficiency of the proposed method drastically and makes it deployable for real-time environment. Moreover, the use of machine learning approach to model the characteristics of byte-level information of benign and malicious packets makes xMiner capable to identify (novel) zero-day attacks.

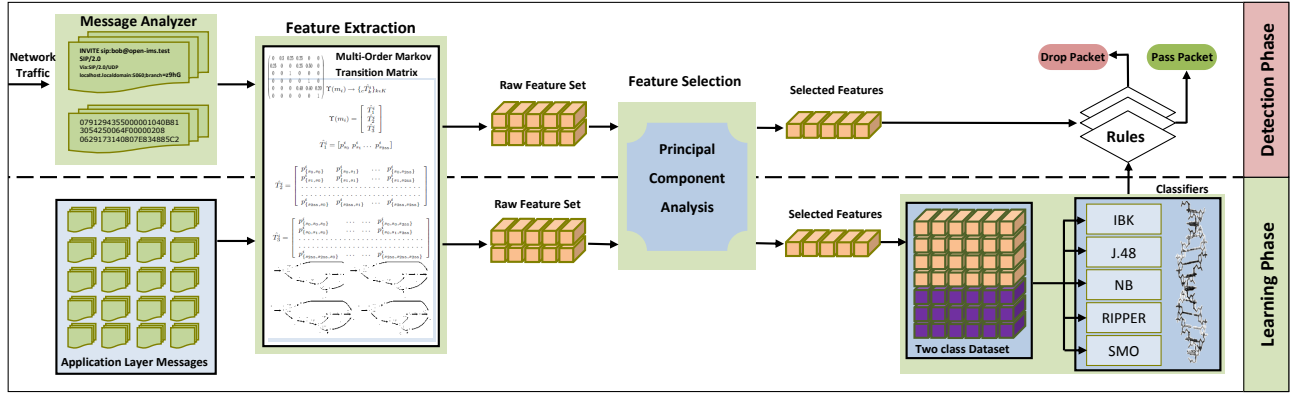


Figure 1. Architecture of xMiner.

III. ARCHITECTURE OF xMINER

In this section, we present the architectural detail of the proposed xMiner to detect vulnerability exploits in different types of application-layer protocols. xMiner is modular in nature and it is developed through analytical research of relevant issues in an engineering fashion. We systematically analyzed different potential solutions and then chose the one capable of meeting our challenges. This modular architecture allows our system to function on different servers and next generation infrastructures. The main functionalities of xMiner are (a) message analysis, (b) feature extraction, (c) feature selection, and (d) model learning and vulnerability exploits detection (see Figure 1). Further details about these functionalities are presented in the following sub-sections.

A. Message Analysis

The main goal during this phase is to analyze network traffics based on the structural formation of the underlying application-specific protocol. The message analyzer module analyzes request and response messages of the underlying protocol to extract discriminative features from them. For network protocols like HTTP, FTP or SIP, packets are captured in form of a raw byte payloads. The structured information of underlying protocols is then acquired by converting raw byte payloads of network traffic using binpac [26]. A sample INVITE request for a SIP protocol, which is used in SIP-based VoIP infrastructure to set up communication session between SIP clients is shown in Figure 2. The overflow of the colons in the second line is mangled to create a buffer overflow exploit. It is important to note that the control information in SIP header is ASCII conforming and contains the necessary information for session setup. The crafty attacker can fuzz different fields in the request message to exploit vulnerability in VoIP servers, which can lead to call processing delays, an unauthorized access or a complete denial of service [8]. It is not possible to predict in advance which fuzzed field can result in a denial of

service attack. Therefore, we have considered the complete incoming messages as an input for our feature extraction module. The analysis is performed on the complete structural formation of the response and request messages.

B. Feature Extraction

A key challenge in extracting features for generic detection of exploits is to make the system adaptable for both text-based and binary protocols. Hence, the tight coupling of feature-set with the formation and syntax of specific protocol is not a feasible choice for our system to operate in diverse environment. Therefore, we have not considered the token-based feature extraction scheme [30] in which special delimiters are used to obtain the features' strings for classification of attack instances in a specific protocol. Rather, xMiner treats a message M_p of a specific application layer protocol P , as an order of elements $\Theta_{M_p} = [\Theta_1 M_p \dots \Theta_l M_p]$, where Θ_{M_p} represents an order of byte values in M and l is the length of the message in bytes. Without loss of generality, we can also incorporate every consecutive n bytes in M_p as a distinctive feature. For example, if $\Theta_{M_p} = [b_1, b_2, b_3, b_4]$, and we consider two consecutive bytes as a feature ($n = 2$), we get the feature-set as $[b_1 b_2, b_2 b_3, b_3 b_4]$. The up-scaling results in higher dimensional feature space of the byte-

```

INVITE sip:bob@open-ims.test SIP/2.0
Via: SIP/2.0/UDP localhost.localdomain:5060;branch=z9hG4bK000000
From: 0; tag=0
To: Receiver
Call-ID: 0@localhost.localdomain
CSeq: 1 INVITE
Contact: 0
Expires: 1200
Max-Forwards: 70
Content-Type: application/sdp
Content-Length: 131

v=0
o=0 0 0 IN IP4 localhost.localdomain
s=Session SDP
c=IN IP4 127.0.0.1
t=0 0
m=audio 9876 RTP/AVP 0
a=rtmap:0 PCMU/8000
  
```

SIP Headers
Session Description Payload

Figure 2. SIP R-DOS INVITE of death.

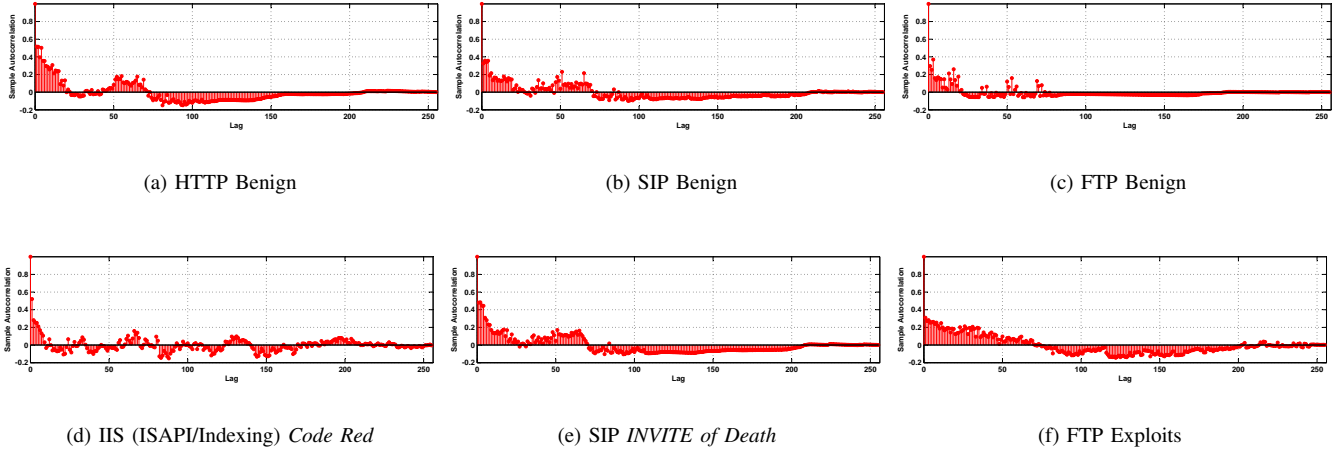


Figure 3. Sample autocorrelation in byte-level sequences of benign and exploit messages.

level distribution of M_p . The large value of n increases the underlying knowledge information and may result in sparse distribution in case the size of training data is not sufficiently large. Alternatively, the small value of n increases the probability of false detection resulting in reduced system performance. Note that the order $n = k$ (where $k > 1$) is simply a joint distribution of byte values with order $n = k - 1$. The up scaling in joint distribution of byte-level features may contain redundant information that are useful for accurate analysis but, at the cost of processing overhead [32]. To this end, we apply number of statistical measures to quantify the order of byte sequence in different protocol messages. A correlation analysis of byte sequences provides valuable insights about the order of their joint distribution. Thereafter, the concept of multi-order Markov process is used to model the information contained in the byte sequences of the messages. These steps are further explained in the following sub-sections.

1) *Correlation Analysis of Byte-Level Sequences*: Autocorrelation is used to study the correlation between the random variables in a stochastic process at different points in time or space. Mathematically, the autocorrelation function of a stochastic process X_z (where z is a time/ space index), for a given lag e , is defined using equation 1 in which $E\{\cdot\}$ is the expected value operator and ρ_{X_i} is the standard deviation of the random variable X_i at time/ space lag z .

$$\rho[e] = \frac{E\{X_0 X_z\} - E\{X_0\}E\{X_z\}}{\rho_{X_0} \rho_{X_z}} \quad (1)$$

The autocorrelation value lies in the range $[-1, 1]$. The $\rho[z] = 1$ means perfect correlation at lag z (which is true for $n=0$) and $\rho[z] = 0$ means no correlation at all at lag z .

To examine the reliance of byte-level sequences in M_p , we calculate sample autocorrelation functions for benign and exploit request messages of different services. Figure

3 shows the sample autocorrelation functions plotted versus the lag of HTTP, SIP and FTP messages. It can be noted in Figure 3 that the byte sequences in request and response messages of different protocols usually follows a 1st, 2nd and 3rd order reliance because the autocorrelation shows peak at $n = 1, 2, 3$ for both benign and exploit messages of all three protocols. This property helps us to model the byte sequences of messages in detecting zero day exploits.

2) *Modeling Byte-Level Sequences using Multi-Order Markov Process*: Application protocols are specified by defined syntax (sequences of bytes) of their request and response messages. Due to dependencies among the bytes of protocol messages, we model the byte-level sequences using Markov chain. The Markov chain uses conditional distribution instead of joint distribution, which results in small sample space, eliminating the redundant information in underlying feature space. A k^{th} order Markov chain of S states can be used to model byte transition probabilities of network messages in a transition matrix T as given in equation 2. In this equation, each state s_i corresponds to byte value b_i . The transition to state s_{i+1} from the state s_i can be computed as $t_{\{s_i, s_{i+1}\}}$ and the probability of state transition as $P_{\{s_i, s_{i+1}\}}$. The underlying assumption is that the probability of each state transition depends only on the previous state values, i.e., in a k^{th} order Markov chain, the value of state s_i can be determined by using the values of the previous states s_{i-k}, \dots, s_{i-1} .

$$T_k = [P_{s_0, \dots, s_k}], \quad (2)$$

A simple (single-order) Markov chain representing the transition probabilities of a byte sequence can be a good choice to model protocol messages when dependencies are homogenous in byte-level sequences. But, in our case the heterogenous nature of byte-level sequences in different protocol messages is evident in the autocorrelation analysis.

Therefore, we argue that the extraction of information from byte-level sequences using a single-order Markov model may not be a perfect choice to model all possible discriminative features necessary for detecting exploits. It should be noted that the order of a Markov chain represents the extent to which past states determine a present state, i.e., how many lags should be examined to analyze higher order sequences [32]. Since, our correlation analysis shows 1st, 2nd and 3rd order dependencies in byte sequences of protocol messages, we have used upto three-order Markov chain to model byte-level information. In this process, we combine transition matrixes \hat{T}_k of multiple orders to achieve more comprehensive modeling of the dependencies in byte-level sequences. The transformation function Υ operates on the bytes of each incoming packet M_p and embeds spatial information in the form of probability values to multiple order state transitions. For a given dataset of Protocol P containing n messages, the multi-order Markov process for i^{th} message in the dataset is computed using equation 3 in which $i = 1, \dots, n$, K is the set of values representing Markov chain orders, and pre-subscript c represents concatenation operator, which combines all \hat{T}_k^i 's into one matrix. For example, the output of transformation function Υ for $K = \{1, 2, 3\}$ can be obtained using equations 4 to 7:

$$\Upsilon(m_i) \rightarrow \{c\hat{T}_k^i\}_{k \in K} \quad (3)$$

$$\Upsilon(m_i) = \begin{bmatrix} \hat{T}_1^i \\ \hat{T}_2^i \\ \hat{T}_3^i \end{bmatrix} \quad (4)$$

$$\hat{T}_1^i = [p_{s_0}^i \ p_{s_1}^i \ \dots \ p_{s_{255}}^i] \quad (5)$$

$$\hat{T}_2^i = \begin{bmatrix} p_{\{s_0, s_0\}}^i & p_{\{s_0, s_1\}}^i & \dots & p_{\{s_0, s_{255}\}}^i \\ p_{\{s_1, s_0\}}^i & p_{\{s_1, s_1\}}^i & \dots & p_{\{s_1, s_{255}\}}^i \\ \dots & \dots & \dots & \dots \\ p_{\{s_{255}, s_0\}}^i & p_{\{s_{255}, s_1\}}^i & \dots & p_{\{s_{255}, s_{255}\}}^i \end{bmatrix} \quad (6)$$

$$\hat{T}_3^i = \begin{bmatrix} p_{\{s_0, s_0, s_0\}}^i & \dots & \dots & p_{\{s_0, s_0, s_{255}\}}^i \\ p_{\{s_0, s_1, s_0\}}^i & \dots & \dots & p_{\{s_0, s_1, s_{255}\}}^i \\ \dots & \dots & \dots & \dots \\ p_{\{s_{255}, s_{255}, s_0\}}^i & \dots & \dots & p_{\{s_{255}, s_{255}, s_{255}\}}^i \end{bmatrix} \quad (7)$$

The transitions in byte sequences incorporates the underlying information from the protocol messages. Any deviation in transition (features) values reflects a different composition of incoming request or response message. This situation indicates an anomalous message, possibly an exploit message, received by the server or in an end device. Since, in the transition matrix not all transitions contain valuable

information to detect exploits messages, we remove them from further consideration to reduce computation overheads.

C. Feature Selection

In this section, we present a discussion about our feature selection process. Having too many features often results in the problem of having too many degrees of freedom leading to poor statistical coverage and thus poor generalization. In addition, each feature adds to a computational burden in terms of processing and storage. Hence, feature selection is an essential step to filter out non-discriminative features from the feature set and thereby to reduce computational overheads. For this purpose, we have used the concept of Principal Component Analysis (PCA) which is a statistical method for dimension reduction. PCA maps high-dimensional data points onto a lower-dimensional set of axes that best explain the variance observed in the dataset.

D. Model Learning and Vulnerability Exploits Detection

Once we have identified the relevant features through applying multi-order Markov process followed by principal component analysis on training dataset, we map every instance of training data into a feature vector to learn classification models. A large body of literatures related to anomaly detection [33], [34], [30] or anomalous network payloads [35], [31], [36], [37] share the common thesis: anomalies are characterized as deviations from a learnt "normal model". In this paper, we have trained five different classifiers – Naïve Bayes (NB), decision tree (J48), inductive rule learner (RIPPER), instance based learner (k-NN), and Support Vector Machine (SVM) using sequential minimal optimization to model the normal behaviors of malicious and benign messages. For model learning we have used Weka³, which is a collection of machine learning algorithms for data mining tasks. Once the classification models are trained they can be easily deployed in real-time to detect messages containing malicious content to exploit vulnerabilities.

IV. EXPERIMENTAL SETUP AND RESULTS

In this section, we present the experimental setup including the dataset description and implementation details of xMiner. The evaluation results of xMiner on different datasets is also presented in this section.

A. Dataset Description

In order to analyze the detection capability of xMiner on real-world attacks launched through various vulnerability exploits, we have collected four real-world traces containing two HTTP, one FTP and one SIP dataset. The first dataset (KSU11) contains 50,000 normal HTTP connections logged from the web server of our institute. The second dataset

³An open source software issued under the GNU General Public License and can be downloaded from the URL: <http://www.cs.waikato.ac.nz/ml/weka/>

Table I
REAL-WORLD EXPLOITS OF HTTP, FTP AND SIP PROTOCOLS.

HTTP							
Id	CVE	Description	Type	Id	CVE	Description	Type
1	2001-0500	IIS (IDQ-Path)	Buf	13	2005-4085	Blue Coat WinProxy Host Header	Buf
2	2001-0241	IIS (Printer Host Header)	Buf	14	2007-0774	Apache mod_jk 1.2.20	Buf
3	2004-1134	IIS(ISAPI w3who.dll Query)	Buf	15	2005-0595	BadBlue 2.5 EXT.dll	Buf
4	2003-0109	IIS (WebDAV intdll Path)	Buf	16	2007-6377	BadBlue 2.72b PassThru	Buf
5	2003-1192	IA WebMail 3.x	Buf	17	2005-3190	CA iTechnology iGateway Debug Mode	Buf
6	2005-2847	Barracuda Remote Command Execution	Web	18	2008-4008	BEA Weblogic Transfer-Encoding	Buf
7	2005-0116	AWStats Remote Command Execution	Web	19	-	cDirectory iMonitor Remote Stack Overflow	Buf
8	2003-0471	Alt-N WebAdmin User	Buf	20	2007-1868	IBM TPM 5.1.0.x rembo.exe	Buf
9	2006-5478	Novell eDirectory (Host Header)	Buf	21	2006-5478	Novell eDirectory Server Host Header	Buf
10	2006-1148	PeerCast <=0.1216 URL Handling	Buf	22	2005-1348	MailEnable Authorization Header	Buf
11	-	HP OpenView NNM-CGI	Buf	23	2004-2271	Minishare 1.4.1	Web
12	2007-5067	Xitami Web Server	Buf	24	2006-5112	NaviCOPA 2.0.1 URL Handling	Buf
FTP							
Id	CVE	Description	Type	Id	CVE	Description	Type
1	2005-0277	3Com 3CDaemon 2.0 FTP Username	Buf	10	2004-0330	Serv-U FTPD MDTM	Buf
2	2006-2961	Cesar FTP 0.99g MKD Command	Buf	11	2005-2373	SlimFTPD LIST Concatenation	Buf
3	2004-2074	BolinTech Dream FTP Server 1.02	Buf	12	1999-0256	War-FTPD 1.65 Password	Buf
4	2006-3952	Easy File Sharing FTP Server 2.0 PASS	Buf	13	1999-0256	War-FTPD 1.65 Username	Buf
5	2006-3726	FileCopa FTP Server pre 18 Jul Version	Buf	14	2006-4318	Texas Imperial Software WFTPD 3.23 SIZE	Buf
6	2006-2407	FreeFTPD 1.0.10 Key Exchange	Buf	15	2004-1135	WS-FTP Server 5.03 MKD	Buf
7	2005-3683	freeFTPD 1.0 Username	Buf	16	2006-3952	Easy File Sharing FTP Server 2.0 PASS	Buf
8	2005-1415	GlobalSCAPE Secure FTP Server	Buf	17	2006-4847	Ipswitch WS_FTP Server 5.05 XMD5	Buf
9	2005-1323	NetTerm NetFTPD USER	Buf				
SIP							
Id	CVE	Description	Type	Id	CVE	Description	Type
1	2009-2867	Cisco 12.2XN(A,B,C,D,T,Z)	R-DoS	10	2003-1112	Ingate Firewall (3.1.3)	RCE
2	2008-3802	Cisco IOS 12.2 through 12.4	R-DoS	11	-	Intelo iGateway-VoIP (1.0.1)	R-DoS
3	2007-4292	Cisco IOS 12.2 through 12.5	R-DoS	12	2003-1113	IPTel OpenSER <=0.8.9	RCE
4	2007-0746	Apple Mac OS X 10.3.9-10.4.9	RCE	13	2003-1109	Cisco IP Phone Model 7940	RCE
5	2007-1306	Asterisk <=1.4.1 && <=1.2.16	R-DoS	14	2003-1114	Mediatix Telecom (SIPv2.4/4.3)	RCE
6	-	INVITE of Death (OpenSBC 1.1.5-25)	R-DoS	15	2003-1109	Cisco IP Phone Model 7960	RCE
7	2007-0961	Cisco PIX 500 and ASA 5500	R-DoS	16	2003-1111	Dynamicsoft products	RCE
8	2003-1108	Alcatel OmniPCX Enterprise 5.0 Lx	RCE	17	2003-1115	Nortel Networks SC Server 2000	RCE
9	2003-1110	Columbia SIP User Agent (sipc)	RCE	18	2006-1973	Linksys RT31P2 VoIP router	R-DoS
10	2005-1461	Ethereal before 0.10.11	Buf				

(CoEIA11) has been collected from the web server of our research center that contains “pure” web application data. This dataset is generated through capturing HTTP traffic for a period of two weeks and contains 37,000 requests.

The FTP dataset is generated by simulating 10,000 clients on Microsoft IIS server with variable user names and passwords. For real-world SIP dataset, we contacted a VoIP vendor that has a customer base in North America. We developed a SIP traffic logger and deployed it on their SIP server and collected a SIP traffic log of more than 20 days containing 4,000 legitimate SIP request and response messages. The log contains traces of SIP dialogs among several SIP terminals as well as SIP dialogs among various network nodes of VoIP infrastructure.

The exploits are collected by using Metasploit framework as well as from common security mailing lists xssed.com, sla.ckers.org and Bugtraq. Some of the exploits used in this study are shown in Table I. We have also used Hzzp fuzzer by *Krakow Labs* [38] to launch attacks through malicious messages on *Microsoft Internet Information Services (IIS)* HTTP server and sniffed 5,000 fuzzed messages through our attack generating machine. This includes request and response fuzzing, authentication fuzzing and query parameter fuzzing. Similarly, we have generated 2,000 fuzzed FTP packets using *FTP Fuzzer* by INFIGO Information Security [39], which is a GUI based fuzzing tool for bench marking the performance of FTP servers. The fuzzing tests covered by the *FTP Fuzzer* unveiled a number of security vulnerabilities (overflows, format strings) in various implementations of FTP servers [40]. Finally, we have used *SIP Security Evaluation Tool* [41] for generating 30,000 malicious SIP

messages. The tool is well known for discovering *INVITE of Death* vulnerability in the SIP stack of an open source SIP server [10]. We have normalized the attack instances by using the similar structure of benign messages to ensure that no obvious artifacts are introduced in attack dataset that makes the detection process intuitively simple.

B. Evaluation Results and Analysis

In our experiments, the purpose is to judge the classification accuracy of xMiner to distinguish between benign and exploit packets. We have done the experiments on an Intel(R) Core(TM) i7 920 @2.67 GHz processor with 12 GB RAM and 64 bit operating system (Windows 7). For each dataset, we have extracted the features using multi-order Markov process and PCA, and then trained five different classification models namely Naïve Bayes (NB), decision tree (J48), inductive rule learner (RIPPER), instance based learner (k-NN), and Support Vector Machine (SVM) using sequential minimal optimization. From the classification results, we calculate the true positive TP (number of malicious packets the system identifies as malicious), the false positive FP (number of benign packets the system identifies as malicious), true negative TN (number of benign packets the system identifies as benign), and the false negatives FN (number of malicious packets the system identifies benign). By using these values we calculate the standard performance measures true positive rate (TPR) and false positive rate (FPR), which are defined in equations 8 and 9. The true positive rate and false positive rate are also called detection accuracy and false alarm rate respectively.

Table II
CLASSIFICATION ACCURACY RESULTS OF xMINER.

	HTTP		FTP		SIP	
	TPR	FPR	TPR	FPR	TPR	FPR
NB	0.977	0.056	0.928	0.064	0.99	0.001
J48	0.994	0.003	0.98	0.001	0.99	0.001
RIPPER	0.992	0.008	0.971	0.029	0.98	0.001
k-NN	0.989	0.004	0.997	0.004	1	0.001
SVM	0.992	0.011	0.936	0.063	1	0
Average	0.9888	0.0164	0.9624	0.0322	0.99	0.0008

$$TPR = \frac{TP}{TP + FN} \quad (8)$$

$$FPR = \frac{FP}{FP + TN} \quad (9)$$

We have used the stratified 10-fold cross-validation for performance evaluation of xMiner, i.e., for each category of data the dataset is randomly divided into 10 smaller subsets, out of which 9 subsets are used for training and 1 subset is used for testing. This process is repeated 10 times for every dataset. For each category of data, the true positive rate (TPR) and false positive rate (FPR) values obtained for different type of classifiers are shown in Table II.

It can be observed from the results presented in Table II that xMiner achieves the detection rate of more than 0.99 for most of the application protocols and even approaching 1 in case of SIP packets. We have also found during experimentation process that the feature selection using PCA improves the detection capability of xMiner. On average, with the help of PCA the TPR is improved by 2% and the FPR is reduced by 3% in detecting malicious exploits of different application services. In our experiments, we have also evaluated the effectiveness of different classifiers on selected features from different datasets. It can be observed from Table II that the TPR and FPR of the decision tree classifier (J48) is optimum for HTTP and FTP packets whereas, the SVM performs well in case of SIP packets with respect to other classifiers.

V. CONCLUSION AND FUTURE WORK

In this paper, we have proposed the design of a generic light-weight vulnerability exploits detection system xMiner to detect malformed packets in real-time. xMiner operates on the byte sequences of network messages and model them using multi-order Markov process. It also applies PCA to reduce the dimensionality of feature space. xMiner is tested on real datasets of HTTP, FTP and SIP messages. On experimentation, we found that xMiner successfully detects exploit messages with average detection accuracy of 99% and false alarm rate of less than 0.1% for different application-specific services. This makes xMiner deployable to protect application servers and devices from real-world threats through effective filtering/blocking of malicious messages. As a result, the threat regarding instant

unavailability of a service can be mitigated. Presently, we are working to enhance xMiner for SMS protocol.

REFERENCES

- [1] H. Wang, C. Guo, D. Simon, and A. Zugenmaier, "Shield: Vulnerability-driven network filters for preventing known vulnerability exploits," in *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4. ACM, 2004, pp. 193–204.
- [2] A. Matrosov, E. Rodionov, D. Harley, and J. Malcho, "Stuxnet Under the Microscope," eset, September 2010. [Online]. Available: http://www.eset.com/resources/white-papers/Stuxnet_Under_the_Microscope.pdf
- [3] "United Press International UPI," January 2009. [Online]. Available: http://www.upi.com/Top_News/2009/01/25/Virus_strikes_15_million_PCs/UPI-19421232924206/
- [4] "Microsoft Security Bulletin MS01-033," November 2003. [Online]. Available: <http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-033.asp>.
- [5] "Microsoft security bulletin ms02-039," January 2003. [Online]. Available: <http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS02-039.asp>.
- [6] "Microsoft Security Bulletin MS03-026," September 2003. [Online]. Available: <http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp>.
- [7] "W32.Sasser.Worm,," April 2004. [Online]. Available: <http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.worm.html>.
- [8] M. Rafique, A. Akbar, and M. Farooq, "Evaluating dos attacks against sip-based voip systems," in *Proceedings of 28th Global Telecommunications Conference (GLOBECOM)*. IEEE, HI, USA, 2009, pp. 1–6.
- [9] C. Mulliner and C. Miller, "Fuzzing the Phone in your Phone," *Black Hat USA*, 2009.
- [10] M. Z. Rafique and M. Farooq, "INVITE of Death, Remote DoS Vulnerability in OpenSBC Server," February 2009. [Online]. Available: <http://ims-bisf.nexginrc.org/OpenSBC-vul.html>
- [11] "Remote DoS on Cisco IOS," 2009. [Online]. Available: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-2867>.
- [12] "Remote DoS on Asterisks SIP Server," 2007. [Online]. Available: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-1306>.
- [13] "Remote Code Execution on Video-Confernce Framework in Apple Mac OS," 2007. [Online]. Available: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-0746>.
- [14] SANS-Institute, "SANS Top-20 2007 Security Risks," 2007. [Online]. Available: <http://www.sans.org/top20/>

- [15] M. Roesch, "Snort-lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX conference on System administration*. Seattle, WA, USA, 1999, pp. 229–238.
- [16] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Comput. Networks*, vol. 31, no. 23, pp. 2435–2463, 1999.
- [17] "Hogwash." [Online]. Available: <http://sourceforge.net/projects/hogwash/>.
- [18] H. Abdelnur *et al.*, "KiF: a stateful SIP fuzzer," in *Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications*. ACM, NY, USA, 2007, pp. 47–56.
- [19] D. Aitel, "An Introduction to SPIKE, the Fuzzer Creation Kit," *immunity inc. white paper*, 2004.
- [20] The-VoIP-Network, "VoIP Market Trends," 2008, <http://www.the-voip-network.com/voipmarket.html>.
- [21] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic classification through simple statistical fingerprinting," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 1, pp. 5–16, 2007.
- [22] "Protecting Internet Data Centers," http://www.arbornetworks.com/dmddocuments/AB_PIDC_EN_Web.pdf.
- [23] S. Niccolini, R. Garroppo, S. Giordano, G. Risi, and S. Ventura, "SIP intrusion detection and prevention: recommendations and prototype implementation," in *1st IEEE Workshop on VoIP Management and Security*, 2006, pp. 47–52.
- [24] V. Apte, Y. Wu, S. Bagchi, S. Garg, and N. Singh, "Space Dive: A Distributed Intrusion Detection System for Voice-over-IP Environments," *DSN 2006*, p. 222.
- [25] D. Geneiatakis, G. Kambourakis, C. Lambrinoukakis, T. Dag-iuklas, and S. Gritzalis, "A framework for protecting a SIP-based infrastructure against malformed message attacks," *Computer Networks*, vol. 51, no. 10, pp. 2580–2593, 2007.
- [26] R. Pang, V. Paxson, R. Sommer, and L. Peterson, "binpac: A yacc for writing application protocol parsers," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. ACM, NY, USA, 2006, pp. 289–300.
- [27] N. Borisov, D. Brumley, H. Wang, J. Dunagan, P. Joshi, and C. Guo, "A generic application-level protocol analyzer and its language," in *14th Symposium on Network and Distributed System Security (NDSS)*, 2007.
- [28] P. Dussel, C. Gehl, P. Laskov, and K. Rieck, "Incorporation of Application Layer Protocol Syntax into Anomaly Detection," in *Proceedings of the 4th International Conference on Information Systems Security*. Springer-Verlag, 2008, pp. 188–202.
- [29] K. Ingham, A. Somayaji, J. Burge, and S. Forrest, "Learning DFA representations of HTTP for protecting web applications," *Computer Networks*, vol. 51, no. 5, pp. 1239–1255, 2007.
- [30] K. Rieck, S. Wahl, P. Laskov, P. Domschitz, and K. Muller, "A self-learning system for detection of anomalous sip messages," in *Principles, Systems and Applications of IP Telecommunications (IPTCOMM), Second International Conference, LNCS*. Springer, 2008, pp. 90–106.
- [31] C. Kruegel and G. Vigna, "Anomaly detection of web-based attacks," in *Proceedings of the 10th ACM conference on Computer and communications security*. ACM, NY, USA, 2003, pp. 251–261.
- [32] F. Ahmed, H. Hameed, M. Shafiq, and M. Farooq, "Using spatio-temporal information in API calls with machine learning algorithms for malware detection," in *Proceedings of the 2nd ACM workshop on Security and artificial intelligence*. ACM, 2009, pp. 55–62.
- [33] S. Forrest, S. Hofmeyr, A. Somayaji, T. Longstaff *et al.*, "A sense of self for unix processes," in *IEEE Symposium on Security and Privacy*. IEEE COMPUTER SOCIETY, 1996, pp. 120–128.
- [34] D. Gao, M. Reiter, and D. Song, "Behavioral distance measurement using hidden markov models," *Lecture Notes in Computer Science*, vol. 4219, p. 19, 2006.
- [35] K. Rieck and P. Laskov, "Language models for detection of unknown attacks in network traffic," *Journal in Computer Virology*, vol. 2, no. 4, pp. 243–256, 2007.
- [36] K. Wang, J. Parekh, and S. Stolfo, "Anagram: A content anomaly detector resistant to mimicry attack," in *Recent Advances in Intrusion Detection*. Springer, 2006, pp. 226–248.
- [37] K. Wang and S. Stolfo, "Anomalous payload-based network intrusion detection," in *Recent Advances in Intrusion Detection*. Springer, 2004, pp. 203–222.
- [38] KrakowLabs, "HTTP compliant client and server fuzzer," <http://www.krakowlabs.com/dev.html>.
- [39] INFOGO-Information-Security, "FTP Fuzzer," <http://www.infigo.hr/files/>.
- [40] J. Leon, "FTP buffer overflow vulnerabilities," <http://www.derkeiler.com/Mailing-Lists/securityfocus/vuln-dev/2006-05/msg00004.html>.
- [41] M. Z. Rafique and A. Yaqub, "SIP Security Evaluation Tool," <http://www.ims-bisf.nexginrc.org/SIPTool.php>.