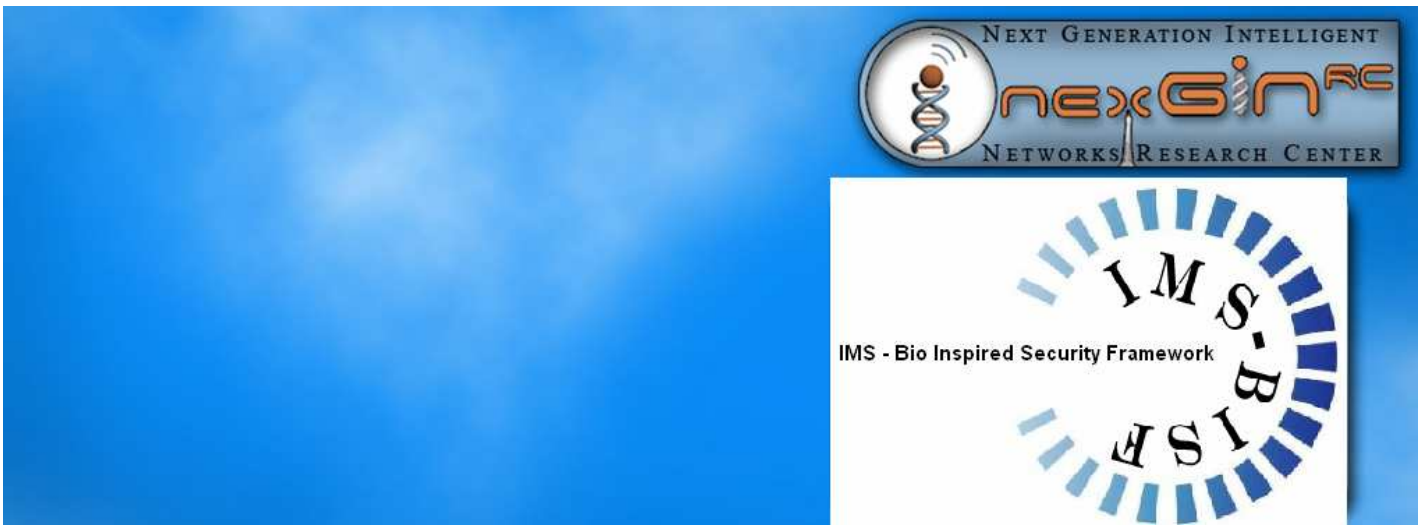
TR-nexGINRC-2010-55



Technical Report

A Bio-inspired Self Defending Security Framework for IP Multimedia Subsystems

(IMS-BISF)

"MESSIAH: Nip the Expolit in the Bud"

18th Mar, 2010

National University of Computer & Emerging Sciences, Islamabad, Pakistan

MESSIAH: To Nip the Exploits in the Bud

M. Zubair Rafique, Muhammad Atif Yaqub and Muddassar Farooq
Next Generation Intelligent Networks Research Center (nexGIN RC)
FAST National University of Computer & Emerging Sciences (NUCES)
Islamabad, Pakistan
Email: {zubair.rafique,atif.yaqub,muddassar.farooq}@nexginrc.org

***Abstract*—Malformed messages in different protocols pose a serious threat because they are used to remotely launch malicious activity. Furthermore, they are capable of crashing – sometimes with a single message only – servers and end points. Recently it is shown that a malformed SMS can crash a mobile phone or gain unfettered access. In contrast, little research work exist to secure servers and mobile phones against malformed messages attacks. In this paper, we incorporate MESsage Syntactical Information to Avoid Hazard (MESSIAH) through malformed messages. We propose a generic malformed message detection framework (MESSIAH) that extracts novel syntactical features from messages/packets of different types of protocols and use them to detect a malformed message. Our framework operates in four steps: (1) analyzing the syntax of a given protocol (say SMS or any other application layer protocol), (2) extracting syntactical features from the messages and representing them in a suffix tree, (3) using well-known feature selection schemes to remove redundancy in the features set, and (4) using standard distance measures to raise the final alarm. The benefit of our framework is that it is lightweight – requiring less processing and memory resources – and provides high detection rate and small false alarm rate. We have evaluated our system on real-world datasets of four protocols: SMS, SIP, HTTP and FTP. The results of our experiments demonstrate that MESSIAH achieves more than 99% detection rate with less than 0.1% false alarm rate. Moreover, its training and testing times for SMS are 2 and 1 milliseconds per SMS respectively. As a result, the framework can be easily deployed on smart phones or mobile devices.**

I. Introduction

It is well known, that remote attackers can cause high profile security threats by exploiting the vulnerabilities in wide range of applications and devices. Almost every remotely launched malicious activity has a seed in a malformed[1] network message. It is shown that a single malformed message can crash a server or endpoint – resulting in complete denial of service. Furthermore, the self-propagating attacks such as worms that exploit vulnerabilities, are launched through malformed messages – an entry point for malicious action. Also a number of security threats – with severity level ranging from moderate to **severe** – launched with the help of malformed messages are documented in [2][3][4][5]. In some scenarios, such attacks result in unacceptable performance; while in worst case scenarios they have successfully crashed a server or an endpoint [1]. A relatively well known example of fuzzed message attack in internet services is a buffer overflow vulnerability, which allow an imposter to inject a shellcode in real time and force the application to execute a malicious code that launches remote shells [6]. Similarly, in *INVITE of Death* attack a malformed packet can launch a remote DoS attack on a real world Voice Over IP (VoIP) server [7].

Recently, it is shown that the extent of damage can exponentially increase when malformed message attacks target smart phones and mobile devices [8]. A recently published survey about emerging smart phone security threats lists fuzzed (SMS) attack [9] a serious emerging threat. The fuzzed SMS attack gives control of instructions – executing within a phone's processor – to an imposter; as a result, he can get unfettered access to the personal information. Moreover, he can execute malicious code that can crash the smart phone. Despite the severity of the problem, malformed message detection has received little attention both in internet services and in mobile phone devices. Mostly researchers follow two approaches for malformed messages detection: (1) signature based intrusion detection, and (2) anomaly detection.

Malformed packet detection techniques using Intrusion Detection System (IDS) methodology is mostly signature based for internet services [10] [11]. It is now a well known fact that signature based techniques cannot cope with exponential increase in new malware because not only the size of signatures database will not scale but the time to match signatures also significantly increases [12][13]. In comparison, non-signature based systems mostly utilize attributes that are specific to each protocol [14] [15]; as a result of this tight coupling between attributive distinctiveness and a particular service make them simply unfeasible for *diverse environments*. Moreover, the problem becomes significantly more challenging in case of resource constrained smart phones [16]. Therefore, we believe that the domain of *generic malformed message detection* is open to novel research.

We incorporated MESsage Syntactical Information to Avoid Hazard (MESSIAH) through malformed messages.

[1]Malformed network messages do not conform to the standard of a protocol and aim to exploit vulnerabilities in the implementation of protocols [1]. In this context of study we use term malformed message, fuzzed message and malicious message interchangeably.

Our proposed framework (MESSIAH) automatically extracts novel *syntactical features* to efficiently detect malformed messages. We pursue the following research methodology in our stagnant analysis: (1) analyzing the syntax of a given protocol (say SMS or any other application layer protocol), (2) extracting syntactical features – which are computable in real time – from the messages and representing them in a suffix tree, (3) using well-known feature selection schemes to remove redundancy in the features set, and (4) using standard distance measures to raise the final alarm. Consequently, our proposed framework consists of four modules: the message analyzer module, the feature extraction module, the feature selection/preprocessing module, and the detection module.

We have evaluated our proposed detection framework on four independently generated malformed message datasets of following services: (1) SMS, (2) VoIP i.e. Session Initiation Protocol (SIP), (3) World Wide Web (WWW) i.e. Hyper Text Transfer Protocol (HTTP) (4) and File Transfer Protocol (FTP). The SMS malformed dataset consists of more than 5000 messages and is generated through SMS Injection framework presented in [8]. For VoIP dataset we have used the SIP Security Evaluation Tool [17] and collected 10000 malformed SIP messages. For HTTP and FTP we have used `hzzp` [18] and `FTPfuzz` [19] malformed packet generating tools. With the help of these tools, we have collected 5000 HTTP and 2000 FTP malformed messages respectively. We have also collected more than 2000 benign SMS messages through our customized *mobile terminal interface* within our social network that consists of people belonging to diverse socioeconomic backgrounds – including engineers, students, housewives, professionals and corporate employees. The real world datasets of SIP have been collected from VoIP service providers; while log of real world HTTP and FTP servers are available at [20] [21].

The results of our experiments show that our framework – MESSIAH – achieves more than 99% detection rate with a false alarm rate $\leq 0.1\%$ for distinguishing between benign and malformed messages for all protocols. Its processing and memory requirements are relatively small; therefore, it can be easily integrated into any of existing servers and devices.

The rest of the paper is organized as following: We present our threat model in Section II. We describe our performance evaluation strategy in Section III that includes the characteristics of our real-world datasets and description of attack injection process in the collected benign traffic. In Section IV, we propose the methodology of our detection framework for fuzzing attacks. We describe the results of our experiments in Section V. In Section **??**, we discuss the limitations and possible extensions of our current approach before discussing related work in Section VI. Finally, we conclude the paper with an outlook to our future work in Section VII.

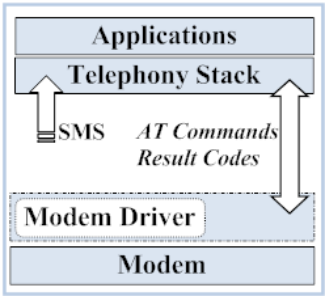


Figure 1. Logical Architecture of Smart Phone

II. Threat Model

We now outline threat models for our targeted services that include SMS, SIP, HTTP and FTP. It is mentioned before that intelligently crafted malformed SMS are being used to launch DoS attacks by exploiting vulnerabilities in the implementation SMS handling modules inside smart phones. The SMS received by the Short Message Service Center (SMSC) on the mobile phone is handled through (Global System For Mobile) GSM modem. (Figure 1 shows the logical architecture of mobile phones.) The GSM modem is controlled through standardized AT[2] commands. Moreover, the modem also provides the interface with GSM network and the application processor of a mobile phone. The SMS received from the modem is delivered to the operating system of the mobile phone through the telephony stack, which has a multiplexing layer that allows multiple applications to access the modem at the same time. Furthermore telephony stack decodes the (Application Program Interface) API-calls into corresponding AT commands and AT result codes to different category messages. An imposter provides the fuzzed SMS as AT result codes to the application processor of a mobile phone in order to trick it to reach an undefined state; as a result, it can lead to significant processing delays, or an unauthorized access or denying legitimate users access to the mobile phone.

Beside smart phones and mobile devices, the malformed message attacks on other services like VoIP and web are pretty common [3] [22] [4] [5]. In these protocols, the underlying flexibility – provided to incorporate new features/services – is most exploited by malformed packets. A common example of malformed packet is Ping of Death (PoD) that sends a malformed Internet Control Message Protocol (ICMP) echo packet which is significantly larger than the maximum IP packet size (64 Kbytes). This attack results in crashing or rebooting of the systems like, Windows 95 and some early NT versions [1]. Similarly, the SIP protocol is based upon an hypertext markup language (HTML) like structure for carrying its requests and responses

[2]AT commands are the de facto standard language for controlling the modems.

[23]. This makes SIP protocol extensible for introducing new features in VoIP. Consequently, making the SIP parser vulnerable against number of unconsidered test cases. The crafty attackers can exploit these vulnerabilities by launching DoS attack through malformed requests at SIP servers. As a result, performance of a SIP server is degraded or it can be crashed (INVITE of Death attack [24]) that ultimately results in unavailability of complete VoIP infrastructure [7].

III. Data Acquisition

We believe that collecting real world datasets for the targeted protocols is critical for evaluating our framework. First, we describe the methodology adopted to collect real world datasets for SMS, SIP, HTTP and FTP protocols. Later, we provide a brief overview of our testbed and accompanying tools that we used for generating malformed datasets .

A. Benign Datasets

We have developed a *modem terminal interface* that can read SMS from the memory of a mobile phone in SMS_DELIVER format. Our interface interacts serially with the modem through *AT commands*. It first configures the modem to operate in *PDU* mode [3] by giving *AT+CMGF=0* command. Once the modem is configured in *PDU* mode, using *AT+CMGL=ALL*, all messages in the memory of mobile phone are redirected to the terminal. We use our *modem terminal interface* to collect the real world dataset in SMS_DELIVER format from people belongs to different socioeconomic backgrounds that include engineers, students, housewives, professionals and corporate employees. This gives us a real world SMS dataset that covers a broad spectrum of messages that can be received by a user on his/her mobile set. We use the NASA's Kennedy Space

Table I
Characteristics of Benign DataSets

Category	Number of Packets	Avg. Size (bytes)	Max. Size (bytes)	Min. Size (bytes)
SMS	2000	225	336	24
HTTP	10000	53	389	8
FTP	2000	13	48	12
SIP	2000	531	1276	495

Center [20] web server log for our HTTP dataset. This is one of the six datasets used by the authors of [25] in there workload study on web servers. We use a subset of this dataset for this purpose. For FTP, we have used the Lawrence Berkeley National Laboratory FTP server log [21]. We have inserted random strings for user name and password in particular FTP commands for evaluating our framework.

For collecting 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. The log contains traces of SIP dialogs among several SIP terminals as well as SIP dialogs among various network nodes of VoIP infrastructure.

[3]PDU (Protocol Description unit) mode is the one in which a modem is configured to read SMS in SMS_DELIVER format.

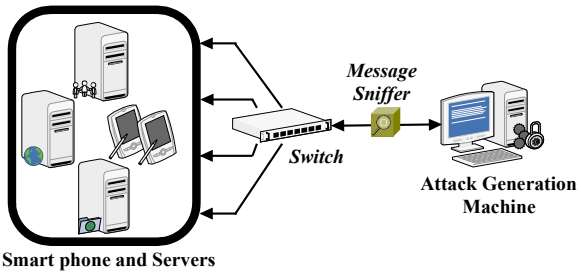


Figure 2. Testbed for generating malformed datasets by launching fuzzing attacks on HTTP server, FTP server, SIP server and Smart phone.

Table II
Characteristics of Malformed DataSets for Randomly Collected 2000 Messages

Category	Number of Packets	Avg. Size (bytes)	Max. Size (bytes)	Min. Size (bytes)
SMS	2000	306	2048	32
HTTP	2000	88	1056	46
FTP	2000	35	1460	30
SIP	2000	620	1872	566

Table III-A shows relevant statistics of benign datasets, which we have used to evaluate the performance of MESSIAH. Note that the size of benign SMS varies extensively from 24 bytes to 336 bytes with an average size of approximately 225 bytes. A similar pattern can be observed for the HTTP, FTP and SIP datasets. The divergence in the size of benign messages is important because mostly large fuzzed messages successfully exploit the implementation related vulnerabilities.

B. Malformed Datasets

We have created a testbed that generates fuzzing attacks targeting smart phones, HTTP servers, FTP servers and SIP servers. The testbed is shown in figure 2. Our testbed consists of *I-Mate Windows Mobile*, *Microsoft Internet Information Services (IIS)* HTTP, FTP [26] server and *OpenSER* [27] (an open source SIP server). The testbed also consists of attack generation machine which is capable of launching fuzzing attacks on the server by utilizing different security tools.

We use the injection framework (*injector*) and Sulley [28] fuzzing framework specially created by the authors of [9] for generating 5000 malformed SMS dataset. The framework allows us to inject the fuzzed SMS into the *iPhone*, *Android*, and *Windows Mobile* devices by sending the SMS over the network through TCP connection on a specified port. We sniff the SMS messages on the network layer and obtain dataset of malformed SMS having different fields

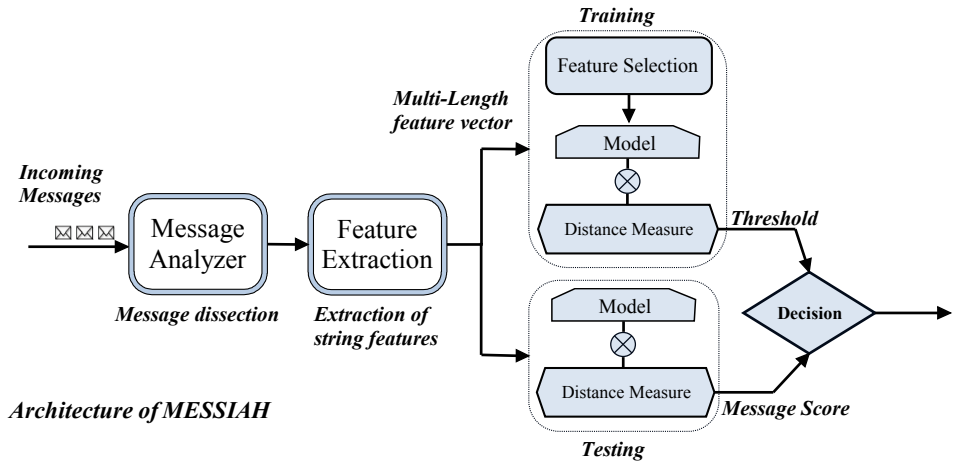


Figure 3. Architecture of MESSIAH: Generic malformed message detection by incorporating syntactical attributes.

(as indicated in table IV-A) fuzzed in a SMS_DELIEVER format. The fuzzed fields can be an SMSC address, sender address, TP-UD etc.

For malformed dataset of HTTP, we have use Hzzp [18] fuzzer by *Krakow Labs*. We have launched attack of malformed message on *Microsoft Internet Information Services (IIS)* HTTP server and sniffed 5000 fuzzed messages – through our *Attack Generating Machine* – that includes the request and response fuzzing, authentication fuzzing and query parameter fuzzing. Similarly, we have used *FTP Fuzzer* [19] by *INFIGO Information Security* – a GUI based fuzzing tool for bench marking the performance of FTP servers – for generating 2000 malformed FTP packets. The fuzzing tests covered by the *FTP Fuzzer* unveiled a number of security vulnerabilities (overflows, format strings) in various implementations of FTP servers [29]. We have obtained a large dataset of malformed FTP messages by launching the test cases through our *Attack Generating Machine* using *FTP Fuzzer* on *Microsoft Internet Information Services (IIS)* FTP server. We have used *SIP Security Evaluation Tool* [17] for generating 10000 malformed SIP messages. The tool is well known for discovering *INVITE of Death* vulnerability in the SIP stack of an open source SIP server [24].

The detailed statistics of the malformed datasets used for evaluating MESSIAH are provided in Table 2. The messages contains numerous fuzzed elements, overflows of the specific string buffers, addition of a large number of token characters and modification of fields in an illegal fashion. Note that the size of fuzzed SMS varies from 32 bytes to 2048 bytes with an average size of approximately 306 bytes. It is important to emphasize that small sized fuzzed messages are difficult to detect compared with the larger ones.

IV. Methodology

In this section, we discuss the architecture and working principle of our proposed malformed detection scheme. We set following requirements for our framework:

- It must be *generic and non-signature based* framework with an ability to operate in diverse environments for wide-range of applications and protocols.
- It should be able to detect both known and novel malformed message attacks.
- It must be deployable on real systems i.e. it should have the detection accuracy of more than 99% with an false alarm rate $\leq$ 1%. Moreover, its time to scan an incoming message must be comparable to those of existing specified techniques for a particular service.
- It should follow the modular design architecture that allows simple yet effective deploying functionality. This feature will be useful in configuring the detection framework for various services.

The final modular architecture of our MESSIAH framework is developed through analytical research of relevant issues in an engineering fashion. We systematically analyzed different potential solutions – capable of meeting our challenges – and then chose the best one. More specifically, we raised following questions and then tried to systematically answer them.

1) Do the format of SMS and other protocol specific messages are suitable for the generalized features' extraction process?
2) Which specific syntactical features can be extracted from incoming network messages to distinguish between benign and malformed messages? Moreover, are the extracted syntactical features better than existing features' set in terms of detection accuracy and efficiency?
3) Do we need to deploy feature selection algorithms on the self generated features' set to generate single model of normality? If yes then which feature selection algorithm is best suited for the raw features' set?
4) Which distance measures or mining algorithm we

can use – to model deviation from normality – for efficient classification of malformed messages with low processing overheads?

The exploration in the design space helped us in designing our malformed message detection framework. It consists of four modules: (1) message analyzer, (2) feature extraction, (3) feature selection and (4) classification (see Figure 3). We now discuss each module separately.

A. Message Analyzer

The core functionality of message analyzer module is to transform the incoming messages into a format from which we can extract intelligent features. We just cite two examples to illustrate our idea.

As mentioned before, SMS messages that are received by the modem through SMSC are in SMS_DELIVER format. Similarly the messages that are sent through mobile phone to SMSC are in SMS_SUBMIT format. Since our framework is designed to operate on local detection (operates on incoming messages) we focus our interest on SMS_DELIVER format. Figure 4 shows the AT result code of receiving a benign and malformed message in SMS_DELIVER format. The first line contains AT Result code and holds the information of the length of SMS. The second line consists of the complete SMS string in hexadecimal representation. The length of SMS in AT Result code is the total number of bytes in the SMS after conversion from hexadecimal to binary. Table 1 shows the partition of SMS_DELIVER message in to the fields along with particular field name, field length and its functionality.

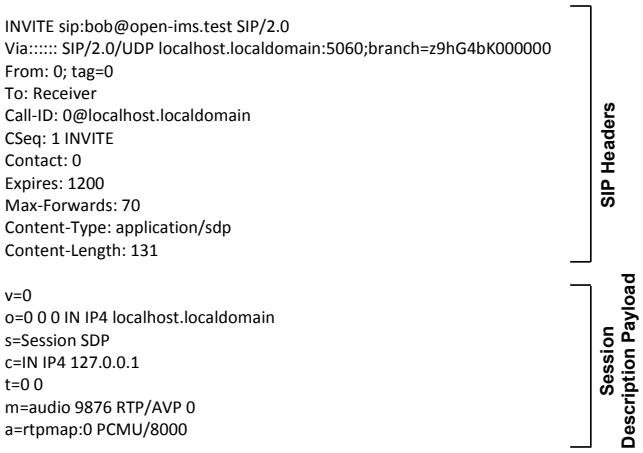


Figure 5. SIP malformed packet `INVITE` of Death

Figure 5 shows the malformed `INVITE` request of a SIP. An `INVITE` message in SIP based VoIP infrastructure is used in setting up communication session between SIP clients. The overflow of the colons in the second line is mangled to create a malformed message. It is important to note that the control information in SIP header is ASCII

Table III
FIELDS OF SMS_DELIVER FORMAT OF BENIGN SMS

Octet(s)	Name	Byte	Functionality
07	AL	1	Length of SMSC Address
91	AT	1	SMSC Address Type
29 43 55 00 00 01	Address	Variable	SMSC Number
04	First Octet	1	Message Indicators
0B	AL	1	Length of Sender Address
81	AT	1	Sender Address Type
30 54 25 00 64 F0	Address	Variable	Sender Number
00	TP-PID	1	Protocol Identifier
00	TP-DCS	1	Data Coding Scheme
20 80 62 91 73 14 08	TP-SCTS	7	Time Stamp
07	TP-UDL	1	User Data Length
E8 34 88 5C 27 97 01	TP-UD	Variable	User Data

conforming and contains the necessary information for session setup. The crafty attacker can fuzz different fields in the request message resulting in a server to reach an undefined state, which can lead to call processing delays, an unauthorized access or a complete denial of service [7]. It is not possible to preditc in advance which fuzzed field can result in the denial of service.

We, therefore, decide to take complete incoming message as a syntactical input for our feature extraction module. For smart phone the message analyzer module extracts the complete SMS message string i.e. the second line of AT Result code and forwards it to the feature extraction module. Similarly for other protocols such as HTTP, SIP and FTP the analysis is performed on the complete syntactical formation of protocol in response/request messages.

B. Feature Extraction

We decided not to predefine the entrenching of incoming messages to syntactical attributes in order to avoid the requirement of coupling features set to a given protocol; as a result, our framework can operate in diverse environments. The motivation for this design option is based on three reasons: (1) Each service/protocol has its specific syntax and structure for the messages, (2) a priory definition of protocol specific attributes can result relatively large memory overheads that make them infeasible solution for resource constraints mobile devices, and (3) a crafty attacker can easily evade a system that works with a set of predefined attributes We have already shown the shortcoming of detecting malformed messages by entrenching incoming messages to syntactical attributes through n-grams based approach [15].

The concept of token characterization also tightly couples features' set with the formation and syntax of an incoming message [14]. The special delimiters are mostly used to obtain the features' strings. For example, Figure 7 shows transformation of a simplified SIP message into a token based feature space using a specific set of predefined delimiters (see [14] for details). It is important to emphasize that the syntax and structure of messages significantly vary for different protocols and services. Figures 5 and 4(a) support this argument by illustrating it for SIP and SMS messages

+CMT: ,34
07912943550000010408813054250064F00000208
0629173140807E834885C279701

(a) Benign SMS message

+CMT: ,42
07912943550000010408813054250064F0000020872
72727272727272727272727272727272727272727272

(b) Malformed SMS message

Figure 4. SMS_DELIEVER format for benign and malformed SMS messages

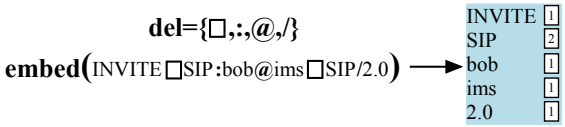


Figure 6. Token based feature space of simplified SIP message

Figure 7. n-gram based feature space of simplified SIP message

respectively. We can now easily conclude that the delimiters used for one particular service can not be utilized for *generic* detection of malformed messages.

If we want to use *n-gram* based feature space then the right choice of the value of n plays a critical role in the system: if n is too small, the probability of false detection increases; otherwise if we choose large value for n, it significantly increases the processing overhead making the framework infeasible for real time systems. The issue is discussed in Rieck et al. in [14] and they have empirically proven that higher values of n-grams performed better but with a trade off in the throughput performance of the system. (A similar trend can be observed in Section V-A.)

Intuitively speaking, for *efficient and generic* malformed message detection, we need a features' set which must meet two requirements: (1) the features space should have the characteristics of higher *n-gram* feature space (to achieve best detection accuracy), and (2) the feature extraction process should have small processing overheads enabling detection in real time. To meet these requirements, we have exploited the properties of Suffix Tree [30] that operates as our entrenching function and it extracts multi-length attributes from an incoming message. A Suffix Tree is a data-structure that is suitable to efficiently mine features from different protocol messages.

We define a set of *attribute strings* 'A' to model the content of an incoming message 'm' from the message analyzer module. We extract multi-length attribute strings A from Suffix Tree, such that for every attribute $a \in A$, the length of a can vary from $a = 1 : |A|$. This eliminates the inherent problem of choosing a suitable value of n – a requirement in *n-grams* characterization. With each *attribute string* 'a' such that $a \in A$, we calculate the number of occurrences of a in m. This gives us an (attribute, value) pair of each attribute $a \in m$ and the value its frequency $f(a, m)$. An entrenching function ψ (Suffix Tree) translates each incoming message m to the $|M|$ dimensional vector $\hat{v}$ space by taking $f(a, m)$ and a in to account that belongs to a particular A.

$$\psi : m \rightarrow \hat{v}^{|M|} \tag{1}$$

with

$$\psi(m) \rightarrow (a, f(m, a))_{a \in A} \tag{2}$$

Figure 8 illustrates the extraction of feature strings and feature space by using Suffix Tree from an incoming message *m*. A Suffix Tree is constructed for each *m* rceived from the message analyzer module. The entrenching function ψ extracts the (attribute, value) pair of features string a and its frequency from the nodes of suffix tree. It can be easily inferred from Figure 8 that the span of entrenching function ψ using suffix tree is equivalent to *n-grams* for all values of n which is greater than *n-gram* for a specific value of n. The authors of [31] also exploited this property of suffix tree to an unsupervised activity analysis. For our *generic* framework that detects malformed messages, the extraction of multi-length features – exhibiting the characteristic of *n-grams'* feature space i.e. $a = 1 : |A|$ – must be linear in time.

We, therefore, have use Ukkonen's algorithm [32] in constructing a suffix tree of each m in linear time. The general method of building suffix tree for any m with length l takes $O(l^2)$ time [33]. In comparison, by using the Ukkonen's algorithm we reduce the time complexity of ψ in mapping the features – into M dimensional vector space – of incoming m with length l to $O(l)$. Moreover, we also reduce the memory requirements by reducing the dimensionality of raw features' set.

We define the set of messages used for training as $K = \{m_1, \ldots, m_k\}$. After each $m \in k$ pass through ψ, we have a set of M dimensional feature vectors as $\widehat{V} = \{\hat{v}_1, \ldots, \hat{v}_k\}$

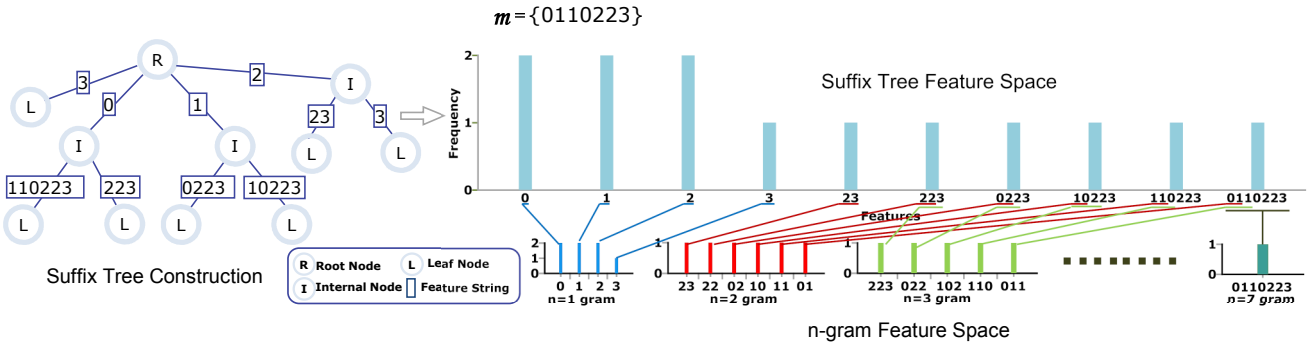


Figure 8. Feature Space of Extracted Attributes Through Suffix Tree

with $\psi(m_i) \rightarrow \hat{v}_i^{|M|}$. In term of (attribute, value) pairs, we have:

$$\widehat{V} = \begin{pmatrix} (a_1, f(m_1, a_1)) & \dots & (a_n, f(m_1, a_n)) \\ (a_1, f(m_2, a_1)) & \dots & (a_n, f(m_2, a_n)) \\ (a_1, f(m_3, a_1)) & \dots & (a_n, f(m_3, a_n)) \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ (a_1, f(m_{k-1}, a_1)) & \dots & (a_n, f(m_{k-1}, a_n)) \\ (a_1, f(m_k, a_1)) & \dots & (a_n, f(m_k, a_n)) \end{pmatrix} \quad (3)$$

Figure 9 shows the frequency comparison of the suffix tree features of malformed and benign SMS messages (see Figure 4). We can see the difference in features' frequency values for benign and malformed messages. Therefore, we can learn a model of "normal" for benign messages and subsequently use some distance measures to detect malformed messages. In order to create "normal model" for our framework – reflecting the vital characteristics of training data – we take the union of feature vectors to obtain raw features' set $\tilde{\mathbf{X}}$ as:

$$\tilde{\mathbf{X}} = \{\hat{v}_1 \cup \hat{v}_2 \cup \hat{v}_3 \dots \hat{v}_k\} \quad (4)$$

To remove the redundant features and increase the efficiency of our framework – both in terms of space and time – we chose different feature selection algorithms. With a given set of training data K and raw model vector $\tilde{X}$, the feature selection algorithm operates on each $m \in k$ and provides a subset $\hat{S}$ of $\hat{X}$ imitating the vital characteristics of training data.

C. Feature Selection

We have now computed the raw feature set $\tilde{X}$ that contains a number of extracted features (from suffix tree) based on syntactical formation of the messages in the training data. It is possible that all features in $\tilde{X}$ might not be useful for detecting malformed messages. Furthermore, the high dimensionality of $\tilde{X}$ will result in large processing

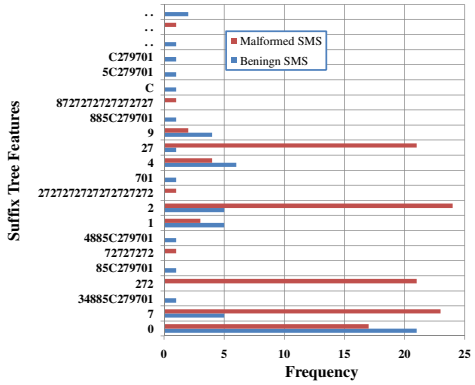


Figure 9. Suffix Tree features comparison of Benign and Malformed SMS

overheads, making the detection process infeasible for resource constraint devices (like mobile phones). Therefore, it is logical to remove redundant features – having low classification potential – to reduce the dimensionality of raw features' set but it should not degrade the false alarm rate. We, therefore, have decided to reduce the dimensionality of raw feature set by ranking the features based on statistics and information theory parameters. This helps us to determine the classification potential of each feature in $\tilde{X}$. We subsequently rank the features as a function of their classification potential and selectt the top-ranked 500 features for classification. We now provide the short description of the techniques that we have used for feature selection.

1) Information Gain: We apply information gain to calculate the classification potential of individual feature in $\hat{X}$. The top-ranked features are obtained by computing the difference in current entropy (improbability) and expected entropy of features in the training data. An interested reader can refer to [34] for a better understanding of information gain.

2) chi Squared: The features in this scheme are selected by measuring the lack of independence between a particular feature in the training data. One can find relevant details about chi squared statistic in [35].

3) Gain Ratio: The gain ratio uses an extra term to cater for symmetrical measure used in calculation of information gain of a feature. This extra term contains information about the training data partitioned by an individual feature. For details about gain ratio, please refer to [36].

D. Classification

Once we have distinct *model set* of attributes ($\hat{S}$) – obtained after reducing the dimensionality of raw features' set – we can easily detect a malformed message by either using well known distance/divergence measures or standard machine learning algorithms. A large body of anomaly detection literature [37] [38] [14] or anomalous network payloads [39][40][41][42] shares the common thesis: anomalies are characterized as deviations from a learnt "normal model" [15]. In our scenario, we can detect malformed messages by transforming incoming message to a suffix tree vector space (i.e $\psi(m)$) and comparing it with $\hat{S}$ – obtained from benign data. We compute K distance scores as $D = \{d_1, d_2, \dots d_k\}$ from incoming messages streams to model deviation pattern of benign messages. (Remember the largest distance value in D still represents a normal message.) Therefore, we need to learn largest threshold value t with the help of largest distance score of a message in the training data K. Any distance value the threshold t is classified as a malformed message. Mathematically, we define the threshold value as:

$$t = max(D_i) \tag{5}$$

where t corresponds to the largest distance value from the model set $\hat{S}$. Using Equation 5, we can raise an alarm just with a single $\geq$ from the threshold value. Furthermore, the calculation of this threshold value is linear in time with complexity $O(K)$. Now the most challenge is to find a suitable distance/divergence measure that provides high detection accuracy and small false alarm rate. We have empirically short listed[4] 6 well known distance measures for our analysis. An interested reader can refer to [43] for a comprehensive review of distance measures.

1) *Canberra Distance* (**cd**) operates on the coordinate pair objects and calculates the series sum of fractional difference between them. Mathematically we represent it as:

$$cd_i = \sum_{\substack{1\leq g\leq|\hat{S}|\\1\leq h\leq|\hat{v}|}} \frac{|\phi(\hat{v}_i(h)) - \phi(\hat{S}(g))|}{|\phi(\hat{v}_i(h))| + |\phi(\hat{S}(g))|} \tag{6}$$

[4]We have use only those distance measures that can operate in high dimensional vectorial representation of Suffix Tree feature space.

Algorithm 1 Distance calculation

```
function DISTANCE(v̂, Ŝ:Dictionary Container)
    d ← 0, g ← 1, h ← 1
    while g ≤ |Ŝ| do
        w← Ŝ(g)
        if v̂ contains att[w] then                ▷ Match Case
            d ← d + F(φ_v(att[w]), φ(w))
            g←g + 1
            remove att[w] in v̂
        else                                     ▷ Mismatch in v̂
            d←d + F(0, φ(w))
            g←g + 1
        end if
    end while
    while h ≤ |v̂| do                             ▷ Mismatch in Ŝ
        v← v̂(h)
        d←d + F(φ(v), 0)
        h←h + 1
    end while
    return d
end function
```

where $\phi(\hat{v}_i(h))$ represents the h^{th} attribute's frequency value in i^{th} message vector. Similarly $\phi(\hat{S}(g))$ represents the value of g^{th} attribute in model set $\hat{S}$. We will stick to these presentations in all subsequent definitions of distance measures.

2) *Manhattan Distance* (**md**) This distance measures the absolute magnitude difference between objects' pair.

$$md_i = \sum_{\substack{1\leq g\leq|\hat{S}|\\1\leq h\leq|\hat{v}|}} |\phi(\hat{v}_i(h)) - \phi(\hat{S}(g))| \tag{7}$$

3) *Bray Curtis Distance* (**bcd**) is the ratio of absolute difference of the values of attributes and the sum of their values. Mathematically it is defined as:

$$bc_i = \frac{\sum_{\substack{1\leq g\leq|\hat{S}|\\1\leq h\leq|\hat{v}|}} |\phi(\hat{v}_i(h)) - \phi(\hat{S}(g))|}{\sum_{\substack{1\leq g\leq|\hat{S}|\\1\leq h\leq|\hat{v}|}} (\phi(\hat{v}_i(h)) + \phi(\hat{S}(g)))} \tag{8}$$

4) *Itakura-Saito Divergence* (**isd**) The divergence is the special form of Bregman distance [43] with a unique convex function.

$$is_i = \sum_{\substack{1\leq g\leq|\hat{S}|\\1\leq h\leq|\hat{v}|}} (\frac{\phi(\hat{v}_i(h))}{\phi(\hat{S}(g))} - \log\frac{\phi(\hat{v}_i(h))}{\phi(\hat{S}(g))} - 1) \tag{9}$$

5) *Kullback - Leibler Divergence* (**kl**) This divergence measures the difference between two distributions.

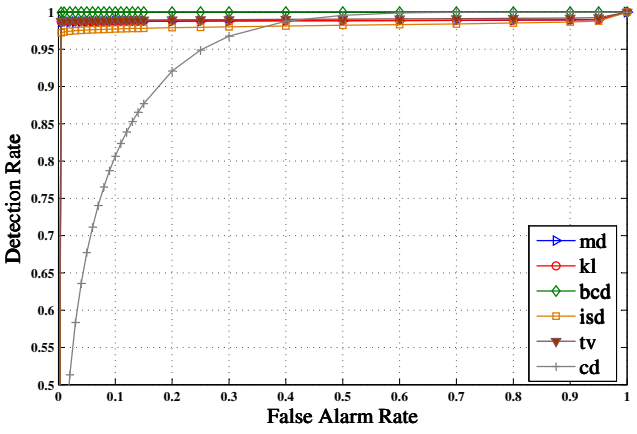


Figure 10. ROC of SMS using Information Gain

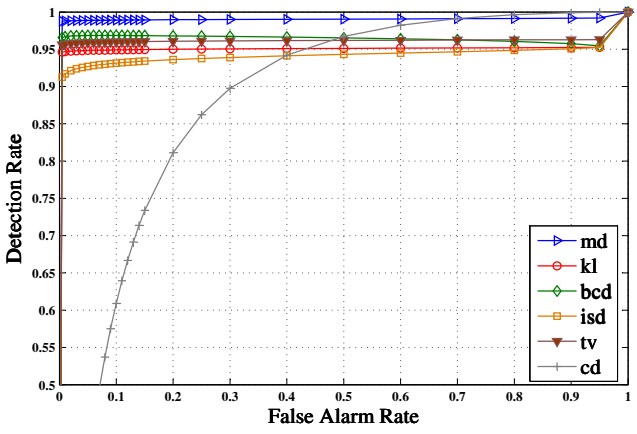


Figure 11. ROC of SMS using Gain Ratio

Mathematically, it is represented as:

$$kl_i = \sum_{\substack{1 \le g \le |\hat{S}| \\ 1 \le h \le |\hat{v}|}} \phi(\hat{v}_i(h)) \log \frac{\phi(\hat{v}_i(h))}{\phi(\hat{S}(g))} \qquad (10)$$

6) *Total Variation* (**tv**) measures the largest probable difference between two series. It is defined as:

$$tv_i = \frac{1}{n} \sum_{\substack{1 \le g \le |\hat{S}| \\ 1 \le h \le |\hat{v}|}} |\phi(\hat{v}_i(h)) - \phi(\hat{S}(g))| \qquad (11)$$

Note that all distance measures have an internal attribute wise function over each dimension that is accumulated over the range of model set and the message vector using $\sum$ operator. We can generalize a distance measure as following:

$$d(\hat{v}, \hat{S}) = \sum_{\substack{1 \le g \le |\hat{S}| \\ 1 \le h \le |\hat{v}|}} F(\phi(\hat{v}(h)), \phi(\hat{S}(g))) \qquad (12)$$

where F represents the inner function of distance measures that operates on incoming message vector $\hat{v}$ and model set $\hat{S}$. For example in case of Manhattan distance the inner function F is $|\phi(\hat{v}_i(h)) - \phi(\hat{S}(g))|$. For efficient calculation of the distance measures we use *dictionary data structure* that is capable of storing objects in sorted order based on multi-length attributes and their frequency values. The distance calculation between two dictionary containers $\hat{S}$ and $\hat{v}$ is in Algorithm 1. The worst case run time of this algorithm (i.e. all comparisons are mismatches) is $O(|\hat{S}| + |\hat{v}|)$, where $|\hat{S}|$ is the length of model vector and $|\hat{v}|$ is the length of message vector. Note that we we have already reduced the length of feature vector (see Section IV-C) and time complexity of algorithm by removing the features from $|\hat{v}|$ in the *Match* case.

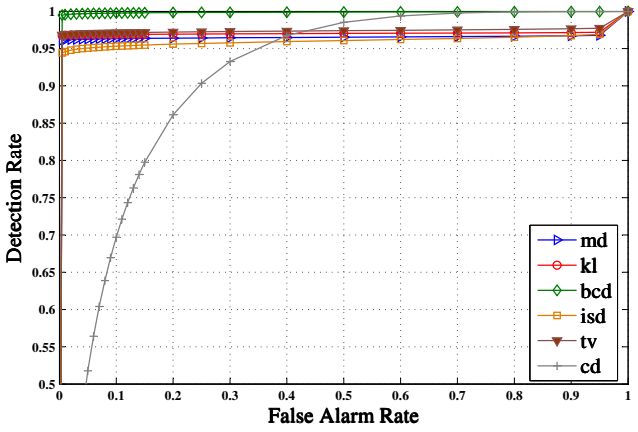


Figure 12. ROC of SMS using Chi Squared

V. Experimental Results And Analysis

In this section, we report the results of our malformed message detector on large number of messages that consist of real world traffic and malformed packets (see Section III). We have designed these experiments to select the best feature selection module and the distance type. We use standard definitions of detection accuracy and false alarm rate with the help of following parameters:

1) Detection of malformed message, true positive (TP).
2) Detection of benign message, false positive (FP).
3) Does not detect a malformed message, false negative (FN).
4) Does not detect a benign message, true negative (TN).

We mathematically define Detection rate or accuracy (DR) and the false alarm rate (FAR) as following:

$$DR = \frac{TP}{TP + FN} \qquad (13)$$

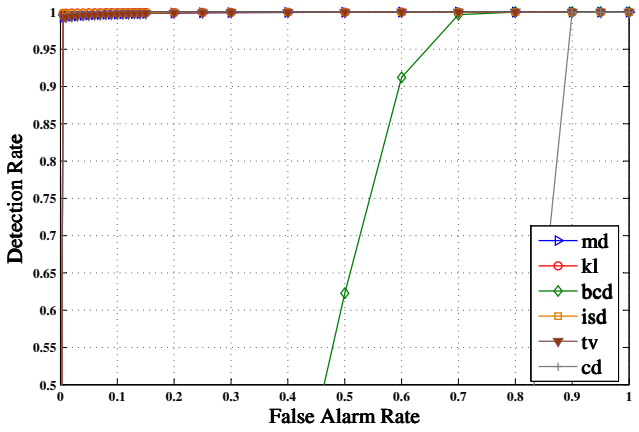


Figure 13. ROC of SIP using Information Gain

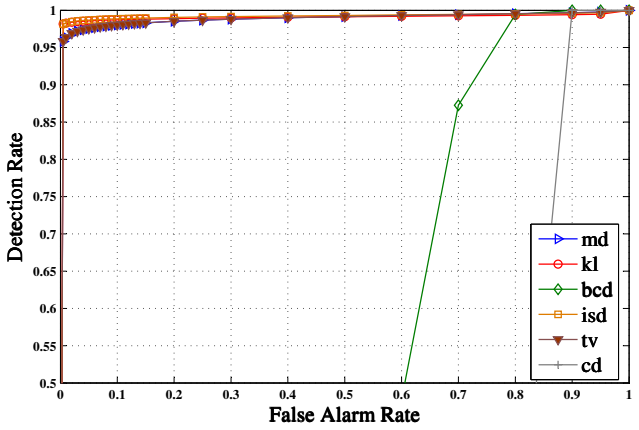


Figure 14. ROC of SIP using Gain Ratio

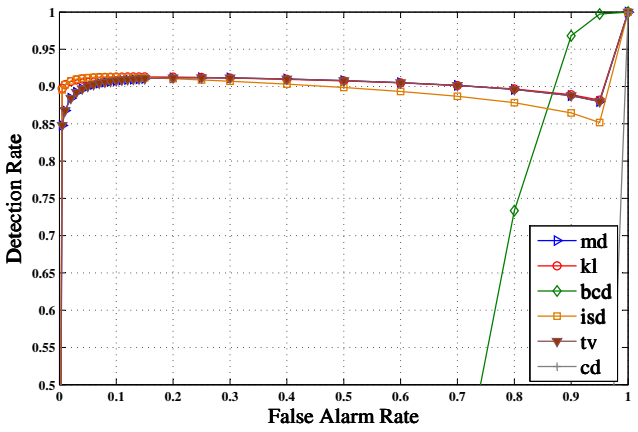


Figure 15. ROC of SIP using Chi Squared

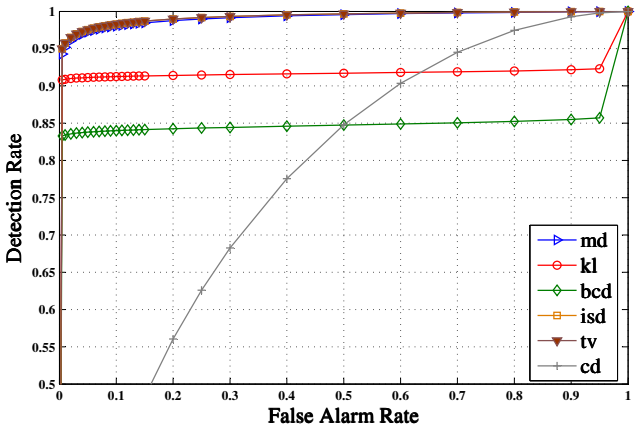


Figure 16. ROC of FTP using Information Gain

$$FAR = \frac{FP}{FP + TN} \tag{14}$$

We train our framework model and calculate the threshold on 100 messages from the benign dataset for different protocols on each distance measure. The model is than evaluated on 500 benign and 500 malformed messages. We repeat the procedure on complete dataset and then report the average results. It is important to emphasize that we use just 100 incoming messages for training phase to learn about "normal model"; as a result, our framework is suitable for mobile devices where SMS traffic is significantly smaller compared with HTTP, FTP or SIP traffic.

We plot ROC curves for different experiment scenarios by varying the threshold on output class probability [44]. The false alarm rate is on the x-axis and the detection rate is on the y-axis for different threshold values of distance measures. The top left portion of the ROC curve reflects high performance while diagonal line reflects the arbitrary detection. We show our results for detection accuracy using suffix tree feature space in Figures 10,16,12,13,19, 14,17,11,18,20,15 and 21. We have also tabulated training and testing overheads for distance measures on different feature selection schemes in Tables IV, V and VI. The training overheads contains the time (in milliseconds) taken by our framework in Feature Extraction (FE), computation of Raw Feature set (RF), building the final model through Feature Selection (FS) and Threshold Calculation (TC) per 100 messages of the training data. All of our experiments are done on an Intel Pentium Core 2 Duo 2.19 GHz processor with 2 GB RAM.

It can be seen from our results that the classification of malformed messages through distance measures – using suffix tree feature space – shows higher detection accuracy on all datasets when features are selected with the help

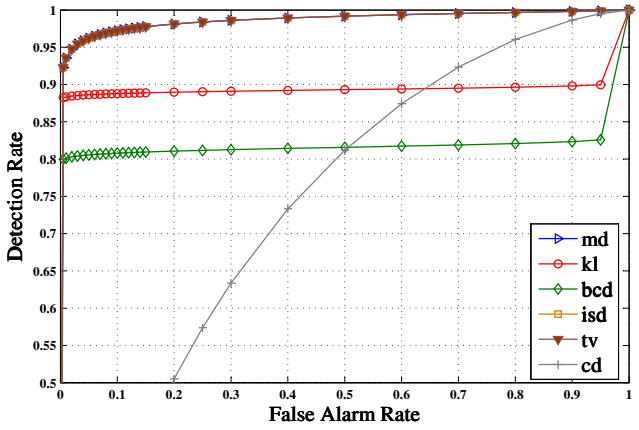


Figure 17. ROC of FTP using Gain Ratio

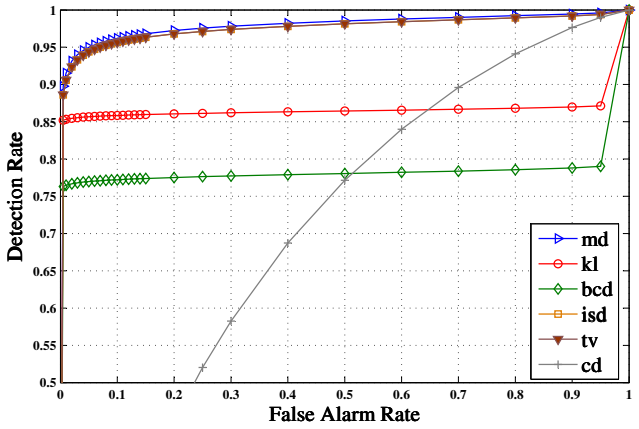


Figure 18. ROC of FTP using Chi Squared

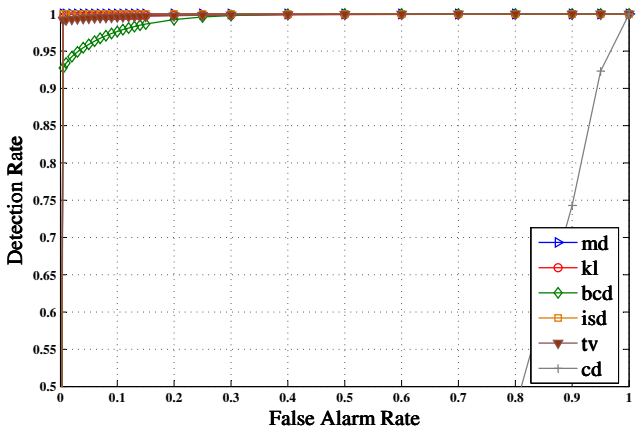


Figure 19. ROC of HTTP using Information Gain

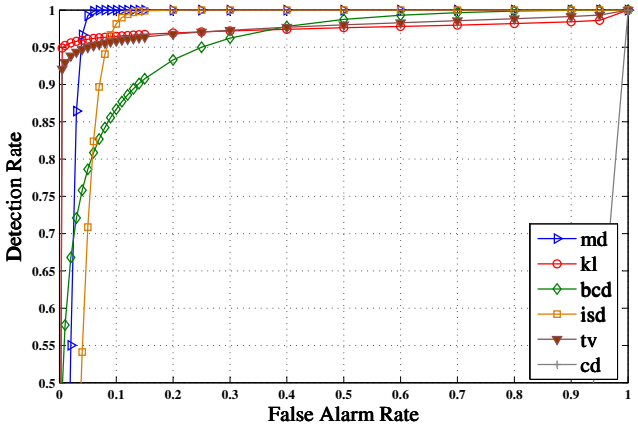


Figure 20. ROC of HTTP using Gain Ratio

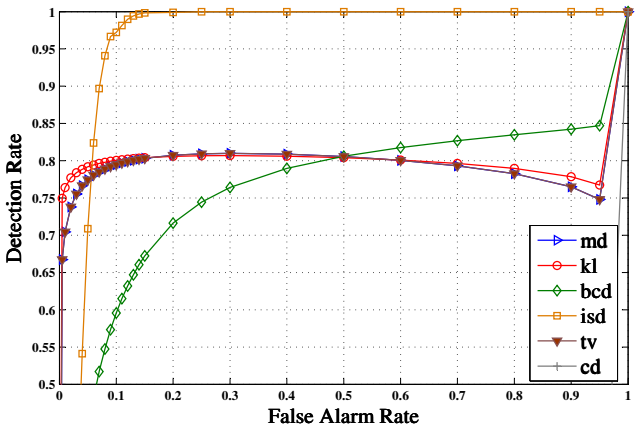


Figure 21. ROC of HTTP using Chi Squared

of information gain. The detection accuracy is worse when features are selected through chi-squared. In particular for SMS dataset, using information gain, the detection accuracy of 99% is achieved for distance measures md, kl, bcd and tv (as shown in Figure 10). Using isd the detection accuracy of our framework is 98% on SMS dataset. The detection accuracy by using cd is not consistent for detecting fuzzed SMS. Similarly on SIP dataset (Figure 13) the detection accuracy of md, kl, isd and tv is 99%. But bcd and cd do not reliably and consistently detect SIP malformed messages even when information gain is used. Similarly, we can see in Figure 16 that the detection accuracy of 99% is achieved using md, isd and tv on malformed FTP dataset. But kl and bcd achieved 90% and 83% detection accuracies on malformed FTP messages. For HTTP (Figure 19) using information gain, the detection accuracy of 99% is achieved for distance measures md, kl, isd and tv. Using bcd the

Table IV
TRAINING OVERHEADS (MILLISECONDS/100MSGS) USING INFORMATION GAIN, GAIN RATIO AND CHI-SQUARED FOR DIFFERENT DATASETS

	SMS			SIP			FTP			HTTP		
	FE	RF	FS	FE	RF	FS	FE	RF	FS	FE	RF	FS
IG	2485	15	47	10687	16	60	31	15	50	63	16	47
GR	2485	15	46	10687	16	60	31	15	47	63	16	46
CS	2485	15	48	10687	16	61	31	15	49	63	16	48

Table V
THRESHOLD CALCULATION IN (MILLISECONDS/100MSGS) USING INFORMATION GAIN, GAIN RATIO AND CHI-SQUARED FOR DIFFERENT DISTANCE MEASURES

	SMS			SIP			FTP			HTTP		
	IG	GR	CS	IG	GR	CS	IG	GR	CS	IG	GR	CS
md	1093	1092	1092	3480	3484	3485	0.03	0.03	0.03	21	20.9	21.1
kl	1093	1093	1093	3500	3500	3500	0.04	0.04	0.04	21.5	21	21.3
bcd	1093	1093	1093	3433	3437	3439	0.04	0.04	0.04	21.8	21.2	21.8
isd	1101	1101	1101	3498	3500	3501	0.03	0.03	0.03	21.8	21.2	21.8
tv	1093	1093	1093	3480	3484	3485	0.03	0.03	0.03	21.2	21	21.2
cd	1020	1019	1019	3433	3438	3437	0.04	0.04	0.04	21.8	21.2	21.6

detection accuracy of our framework for HTTP dataset is just 92%.

In general, if we use gain ratio the detection accuracy decreases for a any distance measure on all datasets as compared to information gain. For instance Figure 12 shows the detection accuracy around 96% using md, tv, isd and kl on SMS dataset using gain ratio. The ROC curves for SIP, HTTP and FTP data sets using gainratio are shown in Figures 14, 20 and 17.

A closer look at the training overheads (see Tables IV, V) reveals no significant difference in processing overhead of different feature selection schemes – gain ratio, information gain and chi-squared. For SMS, our framework took on the average took 3.5 second to train itself on 100 messages. However, this training times depends upon the size of the incoming message. This observation is substantiated in the case of SIP where an average message size is 531 bytes (see Table II). The same pattern is observed in the testing overheads (see Table VI) where the testing time is a function of the size of an incoming message and not a particular feature selection scheme or distance measure. Our framework takes just 10 and 34 milliseconds to test an incoming SMS and SIP message respectively. But for HTTP and FTP messages, the testing overheads are negligible.

The conclusion of our experiments is that manhattan distance and total variation distance measures – in conjunction with the gain ratio feature selection scheme – provides the best detection accuracy, low false alarm rate and small training and testing overheads.

A. *n-gram Feature Space*

In order to evaluate the effectiveness of n-gram approaches, we took 500 features from a SIP packet and used manhattan distance and a number of classifiers to raise the final alarm. We increase the value of n and Figure 22 shows

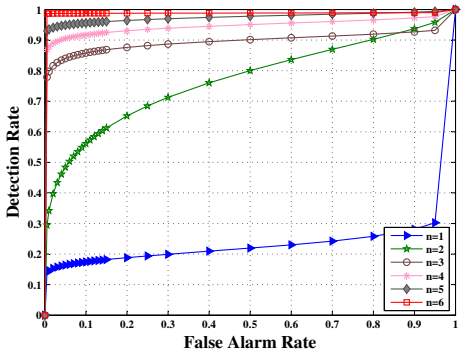


Figure 22. ROC of *n-grams* using Gain Ratio on Manhattan Distance

Table VII
N-GRAMS FEATURE SPACE TRAINING OVERHEADS (MILLISECONDS/100MSGS) FOR SIP USING GAIN RATIO ON MANHATTAN DISTANCE

	n=1	n=2	n=3	n=4	n=5	n=6
FE	218	1172	1781	1968	2188	2391
RF	15	15	15	15	15	15
FS	36	48	50	52	55	57
TC	125	144	2012	2874	3554	4252
TT	394	1379	3858	4909	5812	6715

Table VIII
N-GRAMS FEATURE SPACE TESTING OVERHEADS (MILLISECONDS/1MSG) USING GAIN RATIO FOR MANHATTAN DISTANCE ON SIP

	n=1	n=2	n=3	n=4	n=5	n=6
md	11.2	13.8	20	28.1	34.98	42.5

the evaluation on feature space of $n = 1 \rightarrow 6$ grams. It is obvious from the figure that the detection accuracy improves with an increase in the size of n. But as we increase the size of n, the training and testing times significantly increase (see Tables VII and VIII). Even if we ignore the training overheads, the testing overhead with $n = 6$ in case of SIP is 42 milliseconds. We easily conclude that for a connection rate of just 25 INVITE messages per second, 100% time will be consumed by n-gram malformed packet detector; as a result, server will be left with no time to process INVITE packets. This situation is definitely not acceptable for SIP server. If we extrapolate it to resource constrained mobile phones, the n-gram based approach will simply not work because of the processing overheads.

VI. RELATED WORK

The protection against application level attacks using protcol syntax has been first purposed in signature-based intrusion detection systems. Bro by Paxson [10] – a signature based IDS – uses effective and efficient protocol parsers to detect malformed packets. Similarly, Snort [11] by Roesch is also a signature-based IDS. The signature-based techniques

Table VI
TESTING OVERHEADS (MILLISECONDS/1MSG) USING INFORMATION GAIN, GAIN RATIO AND CHI-SQUARED FOR DIFFERENT DISTANCE MEASURES

	Information Gain						Gain Ratio						chi-Squared					
	md	**kl**	**bcd**	**isd**	**tv**	**cd**	**md**	**kl**	**bcd**	**isd**	**tv**	**cd**	**md**	**kl**	**bcd**	**isd**	**tv**	**cd**
SMS	10.73	10.735	10.735	10.815	10.731	10	10.728	10.735	10.735	10.81	10.735	9.999	10.722	10.739	10.739	10.817	10.739	9.991
SIP	34.6	34.8	34.13	34.78	34.6	34.13	34.64	34.8	34.17	34.8	34.64	34.18	34.65	34.8	34.19	34.81	34.65	34.17
FTP	0.0003	0.0005	0.0005	0.0004	0.0003	0.0005	0.0004	0.0005	0.0005	0.0003	0.0003	0.0005	0.0003	0.0005	0.0005	0.0003	0.0003	0.0005
HTTP	0.2	0.205	0.208	0.208	0.202	0.208	0.199	0.2	0.202	0.202	0.2	0.202	0.201	0.203	0.208	0.208	0.202	0.206

have been used for SIP protocol by Niccolini et al. [45] and Apte et al. [46]. Geneiatakis et al. [47] proposed signatures that prevent fuzzed message attack on VoIP. Most of these techniques are dependant on the signatures' database and hence cannot be used for other protocols specially in the domain of mobile phones. The other signature-based techniques like binpac [48] and GAPAL [49] provide effective and generic procedures of detectiing malformed packets by using protocol parsers. Düssel et al. [15] proposed another approach for the anomaly detection by analyzing the payloads of application-level protocol. The approach detects anomalous packets by computing similarity between the attributed n-grams/tokens derived from the protocol grammar. Ingham et al [50] proposed a learning system on specific presentation by extracting token-based feature through delimiters specific to HTTP requests. A similar work by Rieck et al. [14] proposed the self learning system for detecting malformed messages in SIP. The self learning model operates on the token and n-gram based features. The learned model from higher values of n-gram attributes performed better than the token based attribute model. Kruegel et al. [40] have developed an IDS for HTTP, which uses a number of features like length and character distribution etc to detect malformed packets.

Our detection framework differs from the previous research, as it incorporates suffix tree syntactical feature space to detect malformed messages. Furthermore it does not require any signatures or detection rules for the detection of anomalous event particular to any service. Also we have eliminated the problem of n-gram attribute based model in our feature extraction technique. In particular, the framework operates on the characteristics of received messages from the network and identify malformation attacks for wide range of services having no signatures yet identified.

VII. CONCLUSION AND FUTURE WORK

The major contribution of our work is a generic protocol independent real time malformed message detection framework. Currently, the framework is tested on real datasets of SMS, SIP, HTTP and FTP messages. The results of our experiments show that our framework - using syntactical features - successfully detects malformed messages a detection accuracy of 99% and less than 0.1% false alarm rate. Once we successfully detect malformed messages, we can protect application servers and mobile phones by taking effective flitering/blocking countermeasures; as a result, the threat regarding instant unavailability of a service is mitigated. Our future research focuses on further optimizing the testing overhead of our scheme – specially for SMS and SIP services – and deploying them on real world servers and mobile phones.

REFERENCES

[1] V. Bocan, "Developments in DOS research and mitigating technologies," *Periodica Politehnica, Transactions on Automatic Control and Computer Science*, vol. 49, p. 63, 2004.

[2] K. Corey, "Malformed PPTP packet stream vulnerability," http://www.microsoft.com/technet/security/bulletin/MS01-009.mspx.

[3] "Buffer Overrun In RPC Interface Could Allow Code Execution and Denial of Service," *CERT-In Advisory CIAD-2003-09*, August 2003.

[4] "Multiple vulnerabilities in implementations of the Session Initiation Protocol (SIP)," *CERT(R) Advisory CA-2003-06*, February 2003, http://www.cert.org/advisories/CA-2003-06.html.

[5] "Multiple H.323 Message Vulnerabilities," *CERT(R) Advisory CA-2004-01*, April 2004, http://www.cert.org/advisories/CA-2004-01.html.

[6] R. Verma, "Building a Fuzzing Framework: A Primer for Software Testers," *McAfee Software India*, 2008.

[7] M. Rafique, M. Akbar, and M. Farooq, "Evaluating DoS attacks against SIP-based VoIP systems," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, 2009.

[8] C. Mulliner and C. Miller, "Fuzzing the Phone in your Phone," *Black Hat USA*, 2009.

[9] ——, "Injecting SMS Messages into Smart Phones for Security Analysis," *Third USENIX Workshop on Offensive Technologies*, 2009.

[10] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Comput. Networks*, vol. 31, no. 23, pp. 2435–2463, 1999.

[11] M. Roesch, "Snort-lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX conference on System administration*. USENIX Association, 1999, p. 238.

[12] 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, 2007, pp. 47–56.

[13] D. Aitel, "An Introduction to SPIKE, the Fuzzer Creation Kit," *immunity inc. white paper*, 2004.

[14] 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.

[15] 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, 2008, p. 202.

[16] J. Cheng, S. Wong, H. Yang, and S. Lu, "Smartsiren: virus detection and alert for smartphones."

[17] M. Z. Rafique and A. Yaqub, "SIP Security Evaluation Tool," http://www.ims-bisf.nexginrc.org/SIPTool.php.

[18] KrakowLabs, "HTTP compliant client and server fuzzer," http://www.krakowlabs.com/dev.html.

[19] INFIGO-Information-Security, "FTP Fuzzer," http://www.infigo.hr/files/.

[20] http://www.ita.ee.lbl.gov/html/contrib/NASA-HTTP.html.

[21] http://www.ee.lbl.gov/LBNL-FTP-PKT.html.

[22] J. Fontana, "Exchange Server 5.5 Bug Could Be Exploited for Attacks," November 2000, http://www.pcworld.com/resource/article/0,aid,33882,00.asp.

[23] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: session initiation protocol," RFC 3261, Internet Engineering Task Force, Tech. Rep., 2002.

[24] M. Z. Rafique and S. Aziz, "INVITE of Death, remote denial of serivce vulnerabiltiy in OpenSBC server," February 2009, http://ims-bisf.nexginrc.org/OpenSBC-vul.html.

[25] M. Arlitt and C. Williamson, "Web server workload characterization: The search for invariants," in *Proceedings of the 1996 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. ACM, 1996, p. 137.

[26] Microsoft, "Internet Information Services (IIS)," http://www.microsoft.com/windowsserver2008/en/us/internet-information-services.aspx.

[27] OpenSER, http://www.kamailio.org.

[28] Sulley, "Pure Python fully automated and unattended fuzzing framework," http://code.google.com/p/sulley/.

[29] J. Leon, "FTP buffer overflow vulnerabilities," http://www.derkeiler.com/Mailing-Lists/securityfocus/vuln-dev/2006-05/msg00004.html.

[30] E. McCreight, "A space-economical suffix tree construction algorithm," *Journal of the ACM (JACM)*, vol. 23, no. 2, pp. 262–272, 1976.

[31] R. Hamid, S. Maddi, A. Bobick, and I. Essa, "Structure from statistics: Unsupervised activity analysis using suffix trees," in *In proceedings of International Conference on Computer Vision*, vol. 2007. Citeseer, 2007, p. 1.

[32] E. Ukkonen, "On-line construction of suffix trees," *Algorithmica*, vol. 14, no. 3, pp. 249–260, 1995.

[33] H. Chim and X. Deng, "A new suffix tree similarity measure for document clustering," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, p. 130.

[34] Y. Yang and J. Pedersen, "A comparative study on feature selection in text categorization," in *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*. Citeseer, 1997, pp. 412–420.

[35] A. Moh'd A MESLEH, "Chi Square Feature Extraction Based Svms Arabic Language Text Categorization System," *Journal of Computer Science*, vol. 3, no. 6, pp. 430–435, 2007.

[36] M. Grimaldi, P. Cunningham, and A. Kokaram, "An evaluation of alternative feature selection strategies and ensemble techniques for classifying music," in *Workshop on Multimedia Discovery and Mining*. Citeseer.

[37] 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.

[38] D. Gao, M. Reiter, and D. Song, "Behavioral distance measurement using hidden markov models," *Lecture Notes in Computer Science*, vol. 4219, p. 19, 2006.

[39] 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.

[40] C. Kruegel and G. Vigna, "Anomaly detection of web-based attacks," in *Proceedings of the 10th ACM conference on Computer and communications security*. ACM New York, NY, USA, 2003, pp. 251–261.

[41] K. Wang, J. Parekh, and S. Stolfo, "Anagram: A content anomaly detector resistant to mimicry attack," *Lecture Notes in Computer Science*, vol. 4219, p. 226, 2006.

[42] K. Wang and S. Stolfo, "Anomalous payload-based network intrusion detection," *Lecture Notes in Computer Science*, pp. 203–222, 2004.

[43] T. Cover and J. Thomas, *Elements of information theory*. Wiley, 2006.

[44] T. Fawcett, "ROC graphs: Notes and practical considerations for researchers," *Machine Learning*, vol. 31, 2004.

[45] 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*, 2006, pp. 47–52.

[46] 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.

[47] D. Geneiatakis, G. Kambourakis, C. Lambrinoudakis, T. Dagiuklas, 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.

[48] 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, 2006, p. 300.

[49] N. Borisov, D. Brumley, H. Wang, J. Dunagan, P. Joshi, and C. Guo, "A generic application-level protocol analyzer and its language," in *14h Symposium on Network and Distributed System Security (NDSS)*. Citeseer, 2007.

[50] 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.