Computer Communications 36 (2013) 562–574

Contents lists available at SciVerse ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

SIPAD: SIP–VoIP Anomaly Detection using a Stateful Rule Tree

Dongwon Seo [a], Heejo Lee [a,*], Ejovi Nuwere [b]

[a] Department of Computer Science and Engineering, Korea University, Seoul 136-713, Republic of Korea
[b] SecurityLab Technologies, Tokyo, Japan

ARTICLE INFO

Article history:
Received 31 March 2012
Received in revised form 18 October 2012
Accepted 22 December 2012
Available online 12 January 2013

Keywords:
VoIP security
SIP anomaly detection
Malformed messages
Flooding attacks

ABSTRACT

Voice over IP (VoIP) services have become prevalent lately because of their potential advantages such as economic efficiency and useful features. Meanwhile, Session Initiation Protocol (SIP) is being widely used as a session protocol for the VoIP services. Many mobile VoIP applications have recently been launched, and they are becoming attractive targets for attackers to steal private information. In particular, malformed SIP messages and SIP flooding attacks are the most significant attacks as they cause service disruption by targeting call procedures and system resources. Although much research has been conducted in an effort to address the problems, they remain unresolved challenges due to the ease of launching variants of attacks. In this paper, we propose a stateful SIP inspection mechanism, called SIP–VoIP Anomaly Detection (SIPAD), that leverages a SIP-optimized data structure to detect malformed SIP messages and SIP flooding attacks. SIPAD precomputes the SIP-optimized data structure (termed a stateful rule tree) that reorganizes the SIP rule set by hierarchical correlation. Depending on the current state and the message type, SIPAD determines the corresponding branches from the stateful rule tree, and inspects a SIP message's structure by comparing it to the branches. The SIP-optimized rule tree provides higher detection accuracy, wider detection coverage and faster detection than existing approaches. Conventional SIP inspection schemes tend to have high overhead costs due to the complexity of their rule matching schemes. Experimental results of our SIP-optimized approach, by contrast, indicate that it dramatically reduces overhead and can even be deployed in resource-constrained environments such as smartphones.

1. Introduction

As the Internet is becoming more popular, Voice over IP (VoIP), also called Internet telephony, has become a promising communication medium owing to its economical rates and additional features such as video conversation, SMS, and messenger services. At the same time, VoIP services are facing both known and unknown security threats, as indicated by several studies on VoIP security [1–3].

Two main VoIP session protocols, SIP and H.323, are typically used to establish and terminate a call. SIP is mostly chosen due to its simpler connection process and easier implementation for the Internet [4]. Because of its ease of implementation, the number of SIP–VoIP applications available for mobile phones (e.g., Android and iPhone applications) is rapidly increasing, so that Juniper Research [5] forecasts that the number of mobile VoIP users will reach 640 million by 2016. While SIP–VoIP services are getting popular, they are also becoming attractive targets for attackers who aim to steal private information such as personal contacts and text messages, and to commit fraud by disguising caller information. Recent researches has looked at SIP–VoIP vulnerabilities that can lead to such threats on VoIP services [6,7]. Accordingly, SIP security is an essential part of reliable services provision due to the fact that SIP is in charge of session initiation, connection, and termination.

SIP is particularly susceptible to two types of attacks, malformed SIP message attacks and SIP flooding attacks. Attackers can easily forge the header fields of SIP messages as the messages consists of plain text. Indeed, in the Common Vulnerabilities and Exposures (CVE) [8], 24 malformed SIP vulnerabilities have been publicized between 2010 and the first quarter of 2012. Schanes et al. [9] used a SIP fuzzing framework to find the vulnerabilities in SIP softphones. They found that malformed messages can cause unexpected failures such as program crashes, memory consumption, and process hangs. Consuming resources in a SIP flooding attack is one of the easiest way to crash target servers. In addition, many free downloadable tools can be used to generate SIP messages. These tools not only launch SIP flooding attacks, but also manipulate SIP packets for malformed message attacks [10–12].

Many researchers have previously proposed approaches that detect either malformed message attacks [13,7] or flooding attacks [14–16]. Recently, to cover both types of attacks, Ehlert et al. [17] proposed a two layer SIP prevention mechanism using RFC 3261 [18] rules and a pre-defined threshold. Another approach, Lahmadi

* Corresponding author.
E-mail addresses: aerosmiz@korea.ac.kr (D. Seo), heejo@korea.ac.kr (H. Lee), ejovi@ejovi.net (E. Nuwere).

0140-3664/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comcom.2012.12.004

et al. [19], provides a stateful SIP firewall by generating SIP parsing trees. It checks SIP fields and packet counts using its own pre-defined language. However, all of the approaches have the following drawbacks in terms of detection accuracy and system overhead.

- Low accuracy: Existing approaches are based on SIP standard rules, so-called RFC 3261 rules. However, the standard rules only provide rough boundaries so there is the possibility of being attacked by elaborative malformed messages that meet SIP standard rules but cause unexpected failures by exploiting, for example, string overflow or special character vulnerability. Thus, well-defined SIP rules are necessary to achieve higher detection accuracy.
- High overhead: A SIP message consists of several lines of plain text. Existing approaches to detect malformed messages inspect each line of a SIP header by matching it to pre-defined rules. Matching the header to the rules takes the largest portion of system overhead. Recently, VoIP providers have begun to define their own headers to obtain network or user information; and thus, the number of SIP headers and the inspection overhead increase as the service provides more features. Furthermore, as mobile VoIP services are getting popular, reducing system overhead becomes a critical issue to be feasible in resource-constrained environments such as smartphones.

To detect the two significant attacks with higher accuracy and lower overhead, we propose a novel mechanism, called SIP–VoIP Anomaly Detection (SIPAD), which leverages a `stateful rule tree` based on well-defined SIP rules. From the RFC 3261 SIP standard, we found that there are hierarchical correlations between SIP states and SIP message rules. We used these correlations to design a SIP-optimized data structure, termed the stateful rule tree, to detect the two types of attacks. SIPAD builds tree branches according to the hierarchical correlations between states and rules. For example, the "Confirmed" state allows ACK messages only after receiving an INVITE message. The ACK message should contain mandatory header fields such as Call-ID. By this hierarchical correlation, we can create a stateful tree branch, *Confirmed (state) → ACK (message) → Call-ID (header)*. In addition, we define secure rules that modify RFC 3261 rules to detect elaborative malformed messages. These secure rules refine legitimate cases by concretizing exceptional conditions such as the length of a string and the range of integer values. The stateful rule tree structuralizes correlation branches based on the secure rules so that SIPAD accurately detects anomalies and remarkably reduces rule search complexity. Accordingly, SIPAD can be deployable in resource-constrained environments that have low computational power.

The main contributions of this study are threefold.

1. High detection accuracy: SIPAD refines the original RFC 3261 rules so as to detect elaborate malformed messages. As a result, SIPAD has a 26% higher detection accuracy than that of techniques that use the original RFC 3261 rules.
2. Low system overhead: The stateful rule tree enables fast rule search by constraining the search space. SIPAD's rule search is between 7 and 43 times faster than existing approaches.
3. Wide detection coverage: Because the stateful rule tree defines normal structures for SIP messages, SIPAD detects not only malformed SIP messages and SIP flooding attacks but also three additional anomalies of SIP attacks.

The rest of this paper is organized as follows. In Section 2, we briefly describe the background of SIP communication. Section 3 discusses SIP threat models, and Section 4 defines the problems that we need to solve. In Section 5, we propose a novel mechanism, called SIPAD, that detects SIP attacks, and then, Section 6 analyzes the effectiveness of SIPAD. Section 7 gives an experimental results, and Section 8 introduces related work. Finally, we summarize our results and conclude this paper in Section 9.

2. Background

SIP is in charge of session establishment through the exchange of requests and responses. In order to provide reliable services, it is essential that legitimate messages be transmitted and call procedures upheld. In this section, we briefly give a basic overview of the constitution of SIP messages and its call-setup and tear-down processes.

2.1. SIP messages

A SIP message consists of two parts, a message header and a body. The message header contains essential user information such as Uniform Resource Identifiers (URI), method, and Call-ID. The message body is described as Session Description Protocol (SDP), which is informed for media encoding scheme [20]. Because the message header and body are written in plain text, malformed SIP messages can be easily and diversely generated.

SIP consists of six general requests: INVITE, ACK, BYE, OPTIONS, REGISTER, and CANCEL. INVITE is used to initiate a call to the other party, ACK is a corresponding response to a request, BYE is used to terminate a call, OPTIONS is used to obtain information such as user capability, REGISTER is used to sign in or out from a VoIP provider, and CANCEL is used to abort the latest request. Fig. 1 (left) is an example of a normal INVITE request. Responses (in the form of three digit numbers) are classified into six groups: Provisional (1xx), Success (2xx), Redirection (3xx), Client Error (4xx), Server Error (5xx) and Global Failure (6xx).

The SIP message is processed by a state transition model. In the SIP standard, there are four types of state transition models: INVITE server transition, INVITE client transition, Non-INVITE server transition, and Non-INVITE client transition. When a client receives the INIVTE message, the INVITE server model is activated. On the contrary, when a client sends the INVITE message, the INVITE client model is activated. Similarly, the Non-INVITE server model is activated when a client receives the requests except INVITE, such as ACK, BYE, OPTIONS, etc. Each state transition model includes several states that define the behavior of the SIP procedure. For example, Fig. 2 indicates the INVITE server model with a `Abnormal` state. In Section 5, we describe how the model works in detail.

2.2. The call-setup and tear-down procedure

In order to set up a call, the User Agent Client (UAC) or caller sends an INVITE request to User Agent Server (UAS) or callee as shown in Fig. 1 (right). A proxy server forwards it to the UAS and sends a "100 Trying" response to the UAC. After the UAS receives the INVITE request, it subsequently transfers a "180 Ringing" and a "200 OK" in response. Finally, the UAC receives the "200 OK" response, sends an ACK request and then the connection is established. The UAC and/or the UAS can be malfunctioned if unexpected messages are received during the procedure.

Therefore, malformed and/or unexpected messages are significant threats to SIP systems, and detecting these types of attacks becomes important challenges to provide reliable VoIP services. In Section 3, we introduce SIP threats including the malformed message attacks.

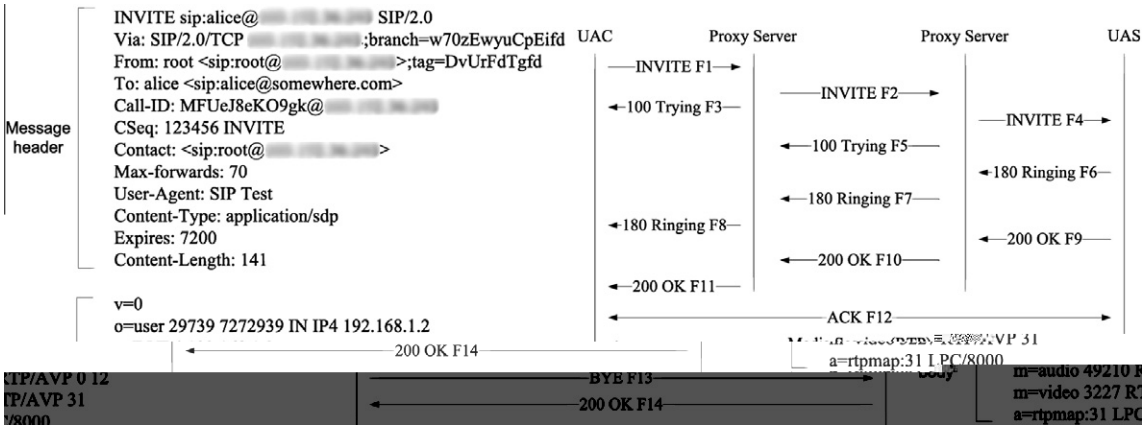

Fig. 1. Normal INVITE request (left) and SIP call-setup and tear-down process (right).

3. SIP threat models

VoIP attacks can be classified into two categories [1,14]: SIP attacks and RTP attacks. SIP attacks exploit session signals (SIP messages) to cause unexpected failures of SIP systems, such as the Denial-of-Service (DoS) and stealing private information, while RTP attacks exploit media packets such as RTP flooding and media spamming attacks [21]. Especially, Keromytis [22] classified the spamming attack, called SPam over Internet Telephony (SPIT), as social threats because it directly targets humans.

Although both SIP and RTP security provisions are significant to provide reliable VoIP services, SIP protection is a preliminary consideration in the sense that SIP is in charge of connection establishment such as session initiation and termination. In this paper, therefore, we focus on the SIP threats, and the followings are SIP related attacks according to Vuong et al. [23] and Ehlert et al. [24].

Malformed message attack. This is one of the most representative cases in which the vulnerabilities of the text-based protocol are exploited. The attack causes proxy servers or User Agents (UAs) to malfunction by manipulating SIP headers. Geneiatakis et al. [6] and Ehlert et al. [24] defined this attack as a SIP parser attack and SIP message payload tampering, respectively, because the attacker intentionally injects forged contents (payload) in the SIP message to cause parsing overhead or even system crash. For instance, SQL code injection can be launched by exploiting the `Authorization` header field [6]. Also, overflow-space, overflow-null, specific header deletion, and the use of non-ASCII code are methods used to carry out in malformed message attacks.

SIP flooding attack. In this type of attack, the attacking tools generate numerous requests or responses to send to specific servers or UAs, so-called victims. As a result, the victim is overwhelmed by excessive SIP messages within a short period of time. Eventually, the victim fails to provide normal services. The SIP flooding attack can be categorized into two types: message flooding and session flooding.

- Message flooding: The attacker sends tremendous SIP messages to specific targets in order for overwhelming their network resources. Ehlert et al. [24] defined this attack as a bandwidth targeting attack. INVITE flooding is one of the most typical flooding attacks. Attackers keep sending INVITE messages to a victim to cause. Furthermore, they modify caller information in order to insert advertisements, and then flood the INVITE messages to random users [21].
- Session flooding: Ehlert et al. [24] also introduced system resource consuming attacks. The victim consumes one's CPU and memory to process many messages. More seriously, the attacker can combine these flooding attacks with the malformed messages to effectively degrade the processing performance.

Spoofing attack. In SIP–VoIP services, two types of spoofing attacks are possible: IP spoofing and URI spoofing attacks. In IP spoofing attacks, the attacker forges IP source addresses and pretends to be a trusted user. While IP spoofing is an attack that is intrinsic to the TCP/IP protocol, URI spoofing is a SIP-based attack caused by malformed SIP messages. An attacker who hijacks SIP messages between two UAs forges their URI field, resulting in the attacker being hidden from tracebacks. Geneiatakis et al. [6] introduced SIP signaling attacks that utilize spoofed signals such as BYE requests. If spoofed BYE requests (BYE signal attack) are sent to a victim, the call will be terminated by the attacker. These vulnerabilities result from lack of authentication. Therefore, SIP over TLS is recently being used to guarantee that authentication takes place.

Snooping attack. This is a passive attack, also called eavesdropping. Attackers only listen, collect, and analyze the passing SIP packets in order to figure out the users' identifiers, media types, network topology, etc. The snooping attack is a fundamental technique used to launch sophisticated attacks such as message interception and modification [22], then the attacker can launch the Man-in-the-middle (MITM) attack on VoIP [25]. As a practical exploitation of the MITM attack, moreover, Zhang et al. [26] showed billing attacks by call session hijacking. Encrypted SIP communication such as SIP over TLS can prevent snooping attacks from succeeding.

Among those threats, malformed messages and SIP flooding attacks take place due to the distinct characteristics of SIP, while spoofing and snooping attacks are derived from IP vulnerabilities. Moreover, these two types of attacks are the major threats in SIP due to the following reasons:

1. Attack target: These two types of attacks do not target specific servers, clients, or software, but protocol vulnerabilities that can affect entire SIP–VoIP systems regardless of service providers. Without addressing these two types of attacks, SIP–VoIP services cannot be reliable.
2. Attack simplicity: While other types of attacks, such as spoofing and snooping attacks need domain knowledge (e.g., protocol stack), these two types of attacks do not require in-depth knowledge. Attacks are just launched by sending many packets and inserting abnormal texts into the SIP fields. Moreover, many tools provide graphic user interfaces (GUI) and allow various input parameters from users; therefore, even "script kids" can easily control the tools and launch variants of an attack.

3. Attack camouflage: Although secure protocols such as SSL/TLS can help to prevent spoofing and snooping attacks, they cannot prevent malformed and flooding attacks, given that the attack messages can be sent through legitimate UAs and treated as normal messages. For example, consider spam e-mails. Regardless of security protocols, spammers can send numerous spam emails because they utilize their one-off user accounts. Similarly, attackers do not have to be concerned about whether or not a VoIP service uses security protocols. They just sign into the VoIP services like legitimate users, manipulate SIP messages, and inject the malformed messages. The VoIP services then treat the malformed messages the same as it does the legitimate ones.

4. Problem definition

This section consists of two parts: problems and goals. We first introduce the problems to be solved in terms of detection difficulty and inefficiency, and then explain the goal of our work.

4.1. Problems

Many approaches have been proposed to defeat the two types of attacks: the malformed message attack and the SIP flooding attack. Most of these approaches address only one of the attacks [13,7,14–16]; although, recently several solutions have been proposed that simultaneously cover both types of attacks [17,19]. However, SIP attack detection needs to address the following challenges in order to be a viable solution:

Difficulty of detecting attack variants. The commonality between malformed messages and SIP flooding attacks is to easily create variants of an attack. Attackers attempt to insert random texts in various fields of a SIP header, and flood with all types of SIP messages. Existing approaches adopt misuse detection based on signatures that cannot cover the numerous variants [19], or consider only the counting of SIP messages to trigger an alert of possible flooding, regardless of the current state of a SIP session [17]. Moreover, existing approaches to detect malformed SIP messages use RFC 3261 rules that provide the basic definition of SIP messages. The approaches merely examine individual lines of a SIP message and so cannot detect structural anomalies. Thus, attackers have sufficient opportunities to create workable variants. A viable solution should propose well-defined SIP rules to detect elaborate malformed messages.

Inefficiency of rule search. Existing malformed message detection schemes find rules from a rule set, and they provide frameworks operating as IDS/IPS and firewall. Therefore, they are suitable for applying to conventional SIP proxy and server sides. These days, however, mobile VoIP applications are dramatically spreading via smartphones, and none of the mobile VoIP applications and devices consider SIP protection. To be deployable to such a resource-constrained environment, a solution should adopt an efficient algorithm.

4.2. Goals

A viable SIP–VoIP detection scheme should address the aforementioned problems, given the increase in threats to SIP and the changes in the user environment, such as mobile VoIP. Therefore, our goal in this work is to design a secure and efficient SIP attack detection scheme based on Anomaly Detection that detects not only the two major types of attacks, but also variant types of attacks with high detection accuracy and low system overhead.

5. SIP–VoIP Anomaly Detection (SIPAD)

In this section, we describe the SIP–VoIP Anomaly Detection (SIPAD) scheme that efficiently detects SIP attacks, including the two most significant types of SIP attacks: malformed SIP messages and SIP flooding attacks.

5.1. SIPAD principle

As mentioned in Section 4.2, we will address the problems, detection difficulty and inefficiency of rule search, and our design principle to address them is as follows.

First, we utilize an anomaly-based detection approach by defining legitimate cases since attackers can easily create numerous variants. Anomaly-based detection has the merit of identifying unknown variant types of attacks. Moreover, it does not need to maintain large amount of attack signatures so that it is suitable for resource-constrained environments. Second, we define a SIP-optimized data structure that can detect SIP anomalies. The optimized structure includes the information to identify whether or not SIP messages accord with the current situation (UA's state) because existing approaches that do not contain the stateful information, such as message exchange procedures, cannot detect elaborate malformed messages or slow rate flooding attacks. Last, we design a low system overhead approach so that it should not disturb the VoIP service. The low system overhead is especially significant for VoIP because it is required to support a real time service.

In this paper, we propose a SIP–VoIP Anomaly Detection (SIPAD) scheme leveraging a SIP-optimized data structure, called a `stateful rule tree`, that applies hierarchical correlations between states and messages from the RFC 3261 standard. The hierarchical correlations allow to construct a tree structure (stateful rule tree) for each state transition model that specifies SIP rule violations. The stateful rule tree inspects the SIP message structure depending on a UA's current state, and provides the optimized rule search path by constraining the search space. Furthermore, SIPAD utilizes well-defined SIP rules and state transition models that precisely define legitimate cases. Therefore, SIPAD can quickly detect variants of an attack in which individual headers meet the RFC 3261 standard but contain structural anomalies.

The most significant issue in achieving these goals is the building of the SIP-optimized structure that detects not only the two major types of attacks, but also variant attacks with high detection accuracy and low system overhead. By analyzing of the RFC 3261 standard, we found the relationship between SIP header fields and states that are linked to specific SIP messages. From the connections between messages, header fields, and current states, we constructed a stateful rule tree that addresses the two issues: detection difficulty and inefficiency of rule search.

In Section 5.2, we describe how SIPAD builds the stateful rule tree. We then explain how SIPAD detects malformed SIP messages and SIP flooding in Sections 5.3 and 5.4, respectively. Finally, in Section 6 we demonstrate the effectiveness of SIPAD.

5.2. Building the stateful rule tree

In the RFC 3261 standard, SIP rules and state transition models are already defined, and existing SIP attack detection approaches utilize this standard description. We also utilize two different detection mechanisms to detect malformed SIP and SIP flooding attacks as follows:

Malformed SIP attack detection. Malformed SIP detection verifies whether or not a SIP message follows pre-defined formats (rules). To apply RFC 3261 rule sets for real VoIP services, we con-

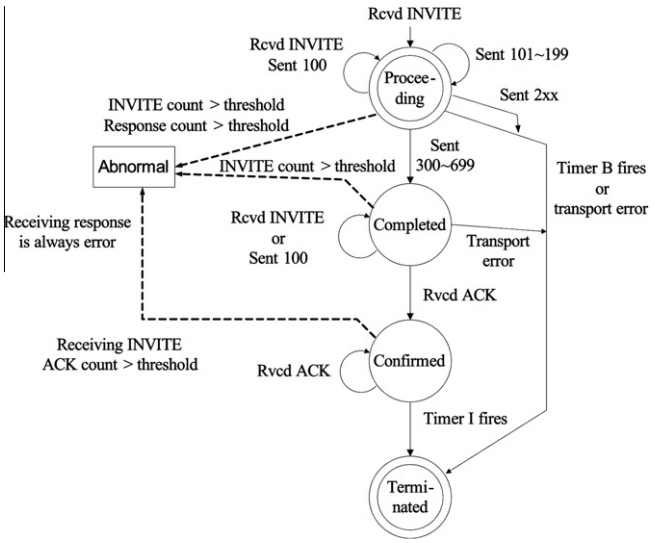

Fig. 2. INVITE server transition model: `Abnormal` state handles flooding condition and improper message transmission.

vert RFC 3261 Augmented BackusNaur Form (ABNF) rules into regular expressions. A rule matching algorithm then decides whether the header of a packet follows its regular expression, and any SIP message that has unmatched or undefined headers is considered a malformed message.

SIP flooding attack detection. We adopted four modified state transition models from RFC 3261,[1] and focused on `INVITE server transition model` to describe how the state transition model works. Fig. 2 illustrates `INVITE server transition model` that is selected when a UA receives an INVITE message. Each state compares the number of received messages with a pre-defined threshold in order to check the flooding condition.

The combination of the two detection algorithms can detect malformed SIP and flooding attacks. However, we found that hierarchical correlations exist between SIP messages, headers, and states as follows:

- A state only links to specific SIP messages.
- A SIP message only links to specific headers.
- A header only links to specific rules.

In Fig. 2, for example, the `Confirmed` state allows ACK messages only. The ACK message only accepts mandatory and optional headers such as "Call-ID" and "From", as shown in Table 1. Then, the headers in the ACK message only accept specific sub-rules such as "callid", "HCOLON", and "from-spec". On the contrary, in the `Confirmed` state, the ACK message that contains the "Accept" header field will be detected as a malformed message. By leveraging the correlations, we specify the rule search path that each state, message and header should follow. That is, we can cut out irrelevant rules while inspecting the message. It allows to detect elaborative malformed messages that follow the irrelevant path, and reduces rule search overhead by narrowing the search space. Therefore, our SIP-optimized structure, called the stateful rule tree, can detect the two types of attacks with high detection accuracy and low overhead.

Fig. 3 illustrates the rule tree for the INVITE server transition model. States, messages and headers are linked depending on the connections. The solid line in Fig. 3 signifies that the `Confirmed` state allows ACK, then ACK allows corresponding headers such as Call-ID and From, and finally, the headers allow corresponding sub-rules such as from-spec, HCOLON, and callid. Thus, the stateful rule tree becomes a hierarchical tree so that a rule can be easily found by retrieving corresponding sub-rules.

The rule tree in SIPAD provides wider detection coverage than existing malformed detecting approaches. For example, Geneiatakis et al. [13] designed a framework based on the RFC 3261 to identify the malformed messages by checking the mandatory fields and the byte size of SIP header and body. However, a malformed message that satisfies mandatory fields but includes non-allowed fields (viz., Table 1) is not detected by the framework since it only checks the syntax of mandatory fields. As a result, SIP proxies and clients can take much more time to process the malformed message and even be crashed while processing the non-allowed fields. Moreover, the framework accepts non-allowed messages regardless of the current state, and this increases the overhead of the SIP system by processing unnecessary messages. For example, in the INVITE server transition model (viz., Fig. 2), although the ACK message is the only one that can be accepted, the framework accepts and processes all types of SIP messages. On the contrary, SIPAD identifies the anomalies of SIP messages using the correlations of header fields and states.

Section 6 analyzes the effectiveness of SIPAD in terms of attack variant detection and rule searching efficiency.

5.3. Detecting malformed SIP message attacks by developing secure SIP rules

We defined the relationship between SIP messages, headers, and states, and constructed a stateful rule tree. Now, defining SIP rules are required to identify that sent/received SIP messages meet to the SIP standard. To achieve the purpose, we adopted RFC 3261 rules. While converting the RFC 3261 rules to regular expressions, we found that the original RFC 3261 rules cannot completely detect malformed SIP messages. For instance, the `userinfo` rule in RFC 3261 that is, **userinfo:((#user#)(:#password#)?)** causes unexpected results such as an overflow exception because it does not check the length of the password. Another example is the port rule. Its ABNF is as follows:

$$port = 1 * DIGIT,$$

where DIGIT denotes a natural number. The corresponding regular expression for the ABNF rule is

$$port = \backslash d+,$$

which means that a port should be a number that consists of more than one digit. Nonetheless, the rule does not check the range of the port number causing overflow-integer. Therefore, we changed the original rule to the more secure one that follows:

$$port = (\backslash d\{0,4\}|[1-5] \setminus d\{4\}|6[0-4] \setminus d\{3\}|65[0-4] \setminus d\{2\}|655[0-2] \setminus d|6553[0-5]), \quad (1)$$

which restricts the port range between 0 and 65535. Thus, if a port number is **65540**, it will conflict with the part of rule (1), in which **655[0–2]\d**. Similar to the example, attackers can make numerous exceptional cases, and lead the malfunction of SIP–VoIP services. For this reason, we design secure SIP rules that constrains the size and format of a string or number. Table 2 shows that several comparisons between regular expressions based on the original RFC 3261 rules and our secure rules. For instance, `user` field allows only alphabet, number, underscore (_), and dash (-). In addition, it must not be more than twelve characters long. Formalizing the SIP standard in this way facilitates the recognition of known and unknown

[1] INVITE server transition, INVITE client transition, Non-INVITE server transition, and Non-INVITE client transition

Table 1
A SIP message has three types of header fields: mandatory, optional and non-allowed. SIPAD categorizes the headers of an ACK message according to the three types.

Types	Header fields
Mandatory (6)	Call-ID, CSeq, From, Max-Forwards, To, via
Optional (13)	Authorization, Contact, Content-Disposition, Content-Encoding, Content-Language, Content-Length, Content-Type, Date, MIME-Version, Record-Route, Route, Timestamp, User-Agent
Non-allowed (25)	Accept, Accept-Encoding, Accept-Language, Alert-Info, Allow, Authentication-Info, Call-Info, Error-Info, Expires, In-Reply-To, Min-Expires, Organization, Priority, Proxy-Authenticate, Proxy-Authorization, Proxy-Require, Reply-To, Require, Retry-After, Server, Subject, Supported, Unsupported, Warning, WWW-Authenticate

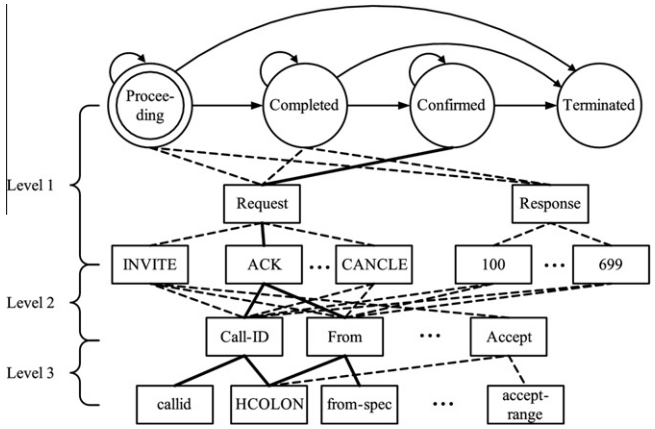

Fig. 3. An example of the stateful rule tree using hierarchical correlations between states and rules: in the Confirmed state, the callee only receives the request, which is ACK that consists of the rules for Call-ID, From, callid, HCOLON, from-spec, etc. (the solid line).

malformed messages. These rules are very flexible and can adopt to a new standard by adding or editing existing rules.

5.4. Detecting SIP flooding attacks by state transition models

As we mentioned in Section 3, SIP flooding attacks can be categorized into two types: message flooding and session flooding. While message flooding targets several specific victims and consume network resources with high rate message generation, session flooding targets numerous session opening and consume system resources (e.g., CPU and memory). Moreover, session flooding can be effective with slow rate message generation.

SIPAD utilizes threshold-base detection to identify the aforementioned flooding attacks. Even though the threshold-based approach is widely used due to its simplicity and it is very suitable for resource-constrained environments, it is required that the threshold should be carefully decided to reduce false positives/negatives. In SIPAD, we adopt two types of thresholds: a state-based threshold and a symmetry threshold. The state-based threshold is to detect the message flooding attack, and the symmetry threshold is to detect the session flooding attack.

State-based threshold. We adopt state transition models to define flooding conditions and improper message transmission. Chen [14] also used the state transition model that defines a threshold for the flooding condition; however, SIPAD enhances the threshold-based detection using the feature of each state. While the existing flooding approach uses a single threshold, SIPAD separates thresholds depending on states. For example, in Fig. 2, the dashed lines signify abnormal conditions for each state. We can separate thresholds depending on states. The INVITE threshold of the `Proceeding` state can be higher than the one of `Complete` state as the caller UA that receives the 300–699 responses does not need to retransmit INVITE messages. Similarly, the INVITE threshold for the `Confirmed` state has to be very low because the caller UA that sends the ACK request should not retransmit INVITE messages.

Symmetry threshold. The existing state-based flooding approaches [14,16] cannot detect the session flooding attack because they only monitor the message generation rate based on the session and do not consider slow rate flooding. If the attacker launches slow rate flooding to avoid the excess of threshold with different session information (e.g., forged call-ID and From header fields), a SIP proxy creates numerous half-open sessions and rapidly consumes its resources. Geneiatakis et al. [27] proposed SIP flooding detection using session distance. They calculated the session distance based on the proportion between the number of INVITEs, ACKs and OK responses. However, the ACK is automatically sent following the OK response, and moreover, it is generated by both a caller and a callee. Thus, the number of ACKs does not affect to the session distance. In SIPAD, we do not count the SIP message that is automatically generated, such as ACK, Trying and Ringing. As a result, in order to detect the session flooding attack, we modify the session distance and define a simpler metric, called SIP symmetry (S). The SIP symmetry is calculated as follows:

$$S = \frac{\text{The number of received SIP requests}}{\text{The number of sent SIP responses}}.$$

SIPAD counts the number of sent and received SIP messages regardless of sessions. Note that we should count the SIP requests and responses that are related to confirming session initiation and establishment and exclude the automatically generated messages. For example, in the Fig. 1 (right), the INVITE message is a request to initiate a session, and the OK message is a response to establish the session. The responses such as Trying and Ringing are automatically sent to the caller, and they are not related to confirming session establishment. In this case, SIPAD divides the number of received INVITE requests by the number of received OK responses. If there is no session flooding attack so that the calling procedure is normally processed, S is close to 1 since the SIP symmetry becomes balanced[2].

Another benefit by using state transition models is to define abnormal conditions according to the states. In the `Confirmed` state, for example, all types of responses and the receipt of INVITE messages are identified as an abnormal condition. In Fig. 4, an example is given of an improper message transmission. When Bob is in the `Confirmed` state, which allows ACK messages only, if Trudy sends INVITE messages, SIPAD detects it as shown. Similarly, we can design the other three state transition models so that it is possible to detect SIP flooding attacks and improper message transmission.

6. Analysis on the effectiveness of SIPAD

In this section, we discuss the types of additional SIP anomalies that can be detected by SIPAD, and then look at how fast SIPAD re-

[2] Note that SIP–VoIP uses the User Datagram Protocol (UDP), and the caller and callee may transmit multiple identical SIP messages to cope with packet loss. From the callee's perspective, as a result, S becomes normally higher than 1.

Table 2
Secure RFC 3261 rules restrain the formats and sizes of strings or numbers (bold fonts) to detect variants of malformed messages.

Original RFC 3261 rules	Secure RFC 3261 rules
user: ((#unreserved#|#escaped#|#user_unreserved#)+)	user: ((#**alphanum**#|_||-){1,12})
password: ((#unreserved#|#escaped#||&||=||+||$ ||,)*)	password: (((#unreserved#|#escaped#||&||=||+||$ ||,)*){0,12})
SIP_Version: (SIP|/|d|.|d)	SIP_Version: ((SIP |/|d|.|d){7,9})
extension_method: (#token#)	extension_method: (#**ASCII_NAME**#{1,20})
protocol_version: (#token#)	protocol_version: (|d{1,2}|.|d{1,2})
display_name: ((#token##LWS#)*|#quoted_string#)	display_name: ((|w){1,32}|#quoted_string#)
callid: (#word#(|@#word#)?)	callid: (**#ASCII#**{1,50}(|@(|w|.*){1,32})?)
Max-Forwards: (Max-Forwards#HCOLON#|d+#CRLF#)	Max-Forwards: (Max-Forwards#HCOLON#|d{1,4}#CRLF#)

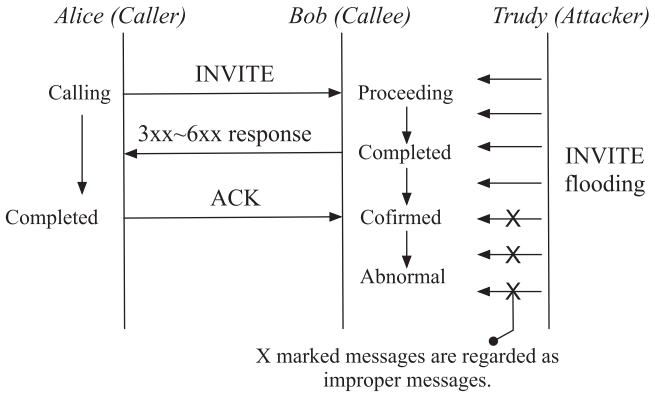

Fig. 4. Improper message transmission: Trudy floods INVITE messages to Bob. Once the Bob's state becomes `Confirmed`, he can detect improper INVITE messages even if Trudy attacks with slow-rate flooding. It is because the `Confirmed` state does not allow all requests except ACK (viz., Fig. 2).

trieves the rules compared to conventional search methods (sequential search and binary search).

6.1. Detecting additional SIP anomalies

An additional benefit of SIPAD is its ability to detect additional SIP anomalies (wider detection coverage). The levels in Fig. 3 indicate the three additional anomalies as follows.

- **Level 1** Improper message transmission: A message is sent/received that is invalid in the current state, as mentioned in Section 5.4.
- **Level 2** Invalid header field: A message includes a non-allowed header field or misses a mandatory header field, as shown Fig. 1.
- **Level 3** Invalid sub-rule: A header filed includes a non-allowed rule based on the RFC 3261 standard.

Because the stateful rule tree in SIPAD is a hierarchical tree so that sent/received messages automatically constrains the paths in the tree, SIPAD makes it possible to detect the additional attacks.

That is, while existing approaches only check whether each part of a message conforms to a corresponding rule, SIPAD additionally checks whether each part of a message conforms to the message structure using the state-based multi-parent rule tree. For instance, assume that a message **A** should consist of **B**, **C** and **D**, expressed as **(A: B C D)**, and a user receives a message **A** consisting of **B**, **C** and **E**, **(A: B C E)**. Because existing approaches look up only the B, C and E rules regardless of the message structure, they cannot identify that **A** is a malformed message. However, SIPAD checks the message structure using the connections between states, messages, and rules and so, as a result, SIPAD can identify that **A** is a malformed message.

6.2. Analysis of rule search efficiency in SIPAD

As shown in Fig. 3, once a SIP message is sent or received, SIPAD limits the state and traversal path, leading to fast rule search. To the best of our knowledge, no study has suggested a SIP rule search algorithm that improves search speed and/or reduces overhead. Therefore, we employ two popular search algorithms, sequential search and binary search, as conventional methods and the stateful rule tree as an improved method. Then, to compare the time complexity, we analyze rule search efficiency using three different versions of SIPAD: SIPAD-Sequential, SIPAD-Binary and SIPAD-SRT (Statreful Rule Tree). The SIPAD-Sequential uses a non-ordered list structure and searches using brute force. On the other hand, the SIPAD-Binary uses a binary search that is sorted in alphabetical order based on the rule's name. the SIPAD-SRT applies the stateful rule tree leveraging the hierarchical correlations between states and rules.

Figs. 5–7 illustrate the major difference among the search methods. For instance, the Call-ID rule consists of as follows:

$$Call-ID : (Call-ID/I)HCOLONcallid.$$

The rule matching engine first finds the Call-ID rule, and recognizes that the header consists of the string "Call-ID" or "I" and two sub-rules, HCOLON and callid. Then, the engine tries to find the two sub-rules, HCOLON and callid, respectively. Here, each method searches in a different way.

SIPAD-Sequential. This method simply search the list from the beginning until finding the HCOLON rule. If found, it searches the list again to find the callid rule. Fig. 5 shows an example of the sequential search. It is the simplest to implement, but the highest time complexity. Furthermore, it does not provide any advantage in terms of SIP attack detection.

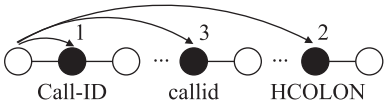

Fig. 5. The sequential search uses a rule list. To find a rule, every time it retrieves the list from the begging to the end.

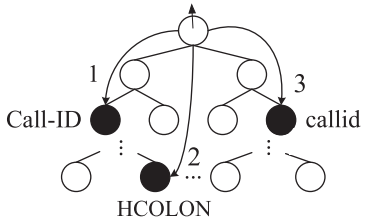

Fig. 6. The binary search uses a alphabetical ordered tree structure. To find a rule, every time it retrieves the tree from the top to the leaf.

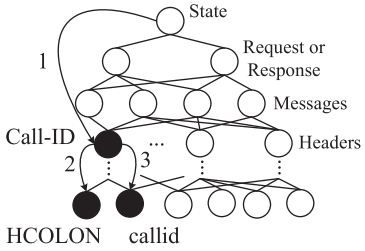

Fig. 7. The stateful rule tree uses a hierarchical tree structure. To find a rule, it retrieves the tree from the top to the leaf once, and then retrieves only the corresponding sub-rule tree.

SIPAD-Binary. This method uses the binary search to the HCOLON rule. Since the binary tree is constructed by alphabetical order of the rule's name, it can reduce the search space. For example, as shown in Fig. 6, it selects a root rule that places in the middle of the rule database, and checks if the selected one is the desired rule (e.g., HCOLON). If not, it determines which rule's name comes earlier in alphabetical order. If the desired one comes earlier than the selected one, the rules located behind the selected one are excluded from further searches. This procedure is repeated until either finding a desired rule or remaining no rule to be selected. However, to find next rule (e.g., callid), the engine should begin searching from the root. In addition, this binary search is not effective for SIP anomaly detection since the rule's name is irrelevant to identifying SIP anomalies.

SIPAD-SRT. This method retrieves the sub-rules from the Call-ID sub-tree, as shown in Fig. 7. Since all the sub-rules related to the Call-ID places in the sub-tree, the engine can greatly reduce the search space to find the HCOLON rule. Moreover, we can identify SIP anomalies in case that a rule does not belong to a proper sub-tree.

To formularize the time complexity of the stateful rule tree, we suppose that,

- n: the number of SIP rules;
- m: the number of sub-rules of a header;
- x: the average number of child nodes of the stateful rule tree;
- k_t: the depth from the root when a sub-rule, t, is found ($1 \leqslant |t| \leqslant m$).

First, SIPAD parses the SIP message, extracts the header field, and searches the header in the stateful rule tree. It takes $O(\log_x n)$. Next, SIPAD finds the sub-rules (m) from the sub-tree of the header. If a sub-rule, t, is found in the kth depth of the tree (k_t), then the stateful rule tree's complexity becomes $O(\log_x n + log_x(n - \sum_{i=1}^{k_t} x^i))$. Because there are m sub-rules to be found, the complexity of the stateful rule tree can be formularized as the following equation:

$$O\left(\log_x n + \sum_{t=1}^{m-1} \log_x \left(n - \sum_{i=1}^{k_t} x^i\right)\right). \tag{2}$$

Now, we can compare the stateful rule tree's complexity with that of the binary search. If $x > 2$, then $log_{10}x > \log_{10}2$ because logarithm is an increasing function. Therefore, the following equation is trivial:

$$\log_2 n - \log_x n = \frac{\log_{10} n}{\log_{10} 2} - \frac{\log_{10} n}{\log_{10} x} = \log_n \left(\frac{1}{\log_{10} 2} - \frac{1}{\log_{10} x}\right) > 0.$$

Finally, we claim that for $x > 2$,

Table 3
The comparison of the time complexity among the sequential, binary and stateful rule tree searches.

Search methods	Time complexity
SIPAD-Sequential	$O(m \cdot n)$
SIPAD-Binary	$O(m \cdot \log_2 n)$
SIPAD-SRT (stateful rule tree)	$O(m \cdot \log_x n)$

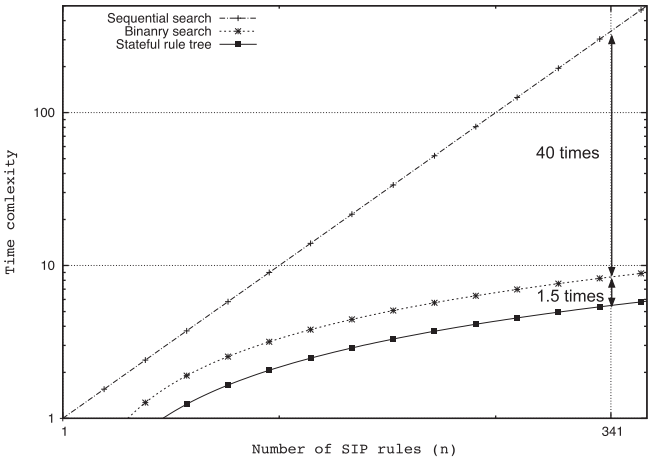

Fig. 8. Given the rules in the RFC 3261 standard, the stateful rule tree shows 62.2 and 1.5 times faster than the sequential and the binary searches, respectively.

$$\log_x n + \sum_{t=1}^{m-1} \log_x \left(n - \sum_{i=1}^{k_t} x^i\right) < \log_x n + (m-1) \cdot \log_x n = m \cdot \log_x n < m \cdot \log_2 n. \tag{3}$$

Eq. (3) indicates that the time complexity of the stateful rule tree is lower than that of binary search. Without loss of generality, we conclude that Eq. (3) does not change if we use $O(m \cdot \log_x n)$ instead of the complex Eq. (2). Table 3 summarizes the time complexity between the sequential, binary, and stateful rule tree searches, when there are m sub-rules in a header.

Furthermore, from analysis of the rules in the RFC 3261 standard, we obtain $n = 341$ and $x = 2.9$ to examine the time complexity depending on the number of sub-rules to find. The result is graphically illustrated in Fig. 8. The binary search is 40 times faster than the sequential search, and the stateful rule tree is 1.5 times faster than the binary search. Considering that the number of rules increases due to implementing various features (e.g., account synchronization) for VoIP services, we expect the performance gap will increase. For example, assuming that x is the same and $n = 1000$, the stateful rule tree is 154 times faster than the sequential search. While this result shows approximate time complexity based on the average number of rules, in Section 7.3, we will implement the three different search methods using 341 SIP rules, and compare the effectiveness of the stateful rule tree in terms of rule search count and time.

In summary, we have shown that SIPAD detects not only two significant types of SIP attacks but also three additional types of SIP attacks. In addition, SIPAD significantly reduces rule search overhead so that it is possible to utilize it in resource-constrained environments such as mobile VoIP services.

7. Experimental results

This section gives experimental results to show the extent to which SIPAD increases detection accuracy and efficiency. First,

we evaluate detection accuracy for two major types of attacks: malformed SIP messages and SIP flooding attacks. We then examine the system overhead for different rule search methods in terms of search count and time. Finally, we summarize the advantages of SIPAD compared to existing approaches.

We have implemented two versions of SIPAD: one for Windows and the other for Android, which are the most popular operating systems for PCs and smartphones. The smartphone environment, in particular, provides the best environment to evaluate SIP security mechanisms as it has restricted resources (e.g., low computational power and memory) and provides many mobile VoIP applications.

Our experimental environment was as follows:

- PC: Intel Core2duo 2.13 GHz, 3 GB RAM, Windows 7, Visual studio 2008.
- Smartphone: Google developer phone (Nexus one), Qualcomm Snapdragon 1 GHz, 512 MB RAM, Android 2.3, Android SDK.

7.1. Malformed SIP message detection results

In the malformed SIP message detection experiment, we used publicly available attacking tools such as PROTOS [28] and SiVuS [10]. PROTOS is a popular VoIP vulnerability assessment tool and PROTOS Test-suite:c07-sip provides 4527 malformed SIP messages. The PROTOS suite has been widely used and is publicly available for use in the evaluation of the implementation level security and robustness of SIP implementations. SiVuS is also a free VoIP vulnerability scanner that has the ability to generate forged packets by editing SIP header fields.

A subset of SIP from the PROTOS suite, namely INVITE messages, was chosen as the subject protocol for vulnerability assessment through syntax testing and test-suite creation. An exceptional element is a piece of data designed to provoke undesired behavior of the test subject.

Among the 4527 test cases of malformed SIP messages, 2426 are associated with malformed SIP message header, and the PROTOS suite provides command-line interfaces to transmit specific malformed cases. SIP exceptional cases are categorized in Table 4. While inspecting the PROTOS exceptional cases, we found that there are a number of ambiguous cases in the middle of valid forms and invalid forms. For example, `aaaaa@sip.invalid.com` can be a valid URI form, but it is identified as an exceptional case in the PROTOS suite. Thus, we redefined such 217 cases as legitimate ones, bringing the total exceptional cases to 2209. When the original RFC 3261 rules were applied, 1837 of 2209 (74%) exceptional cases were detected as malformed messages, while our secure SIP rules detected 100% of them. Fig. 9 depicts the number of exceptional cases detected versus the total number of exceptional cases. The *x*-axis in Fig. 9 is the same as the case number in Table 4.

Furthermore, to verify whether existing VoIP services follow SIP specification, five SIP softphones were chosen from "myvoipprovider.com" web site [29], which offers the top 100 ranking of 155 international VoIP providers. We selected five softphones providing free PC to PC VoIP services based on SIP: Globe7, Vbuzzer, VoIPGo, Gizmo Project and SJPhone.

While testing the existing VoIP services, SIPAD detected several SIP headers that can cause VoIP security vulnerabilities. Each VoIP service had additional specific header fields such as `PortaBilling` for billing information in Globe7. Vbuzzer was also using `Warning` header fields to transmit noisy feedback. Gizmo Project defined extra header fields, such as `JabberID`, `CQBM` and `RemoteIP`. VoIPGo used a format string when there is a space in a username. For example, if a username is `voip go`, it is going to change to `voip%20go` because 0×20 is the ASCII code for the space character. Format string is also included PROTOS exceptional

Table 4
SIP exceptional cases in PROTOS test suite.

#Case	Exceptional elements	Descriptions
1	Overflow-general, space and null	Repetition of general character, space or null
	Format string	Using format string. Ex) %s%d%f
	UTF-8	UTF-8 code. Ex) Chinese characters
	ANSI-escape	Start with characters ESC (ASCII 27d/ 1Bh/ 033o) and [(left bracket)
2	SIP-URI	Invalid SIP-URI form. Ex) sip: aaa:bbb@ccc.ddd, port number should be a number not character like "bbb"
3	SIP-version	"SIP" must have existed. Ex) SIP:2.0
4	IPv4-ASCII	The number range should be from 0 to 255
5	Integer-ASCII	The number ranges are needed. Ex) port number
6	Overflow-colon	Only one colon is allowed. Ex) sip:::::invalid.com
7	SIP-tag	Only one semi-colon is allowed for any option tag Ex) ⟨sip:⟨From⟩⟩;;;;=token
8	Overflow-bracket	Only one bracket (⟨or⟩) is allowed
9	Overflow-at	Only one at (@) is allowed
10	CRLF (Carriage Return/Line Feed)	Every single line should have only one CRLF at the end

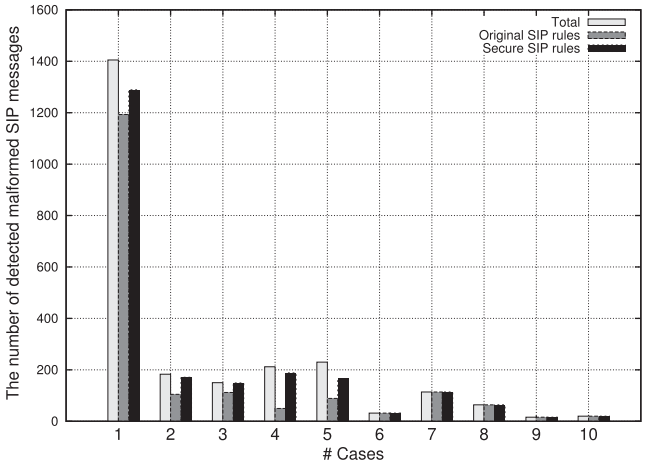

Fig. 9. Secure rules detect 25% more PROTOS exceptional cases than original rules.

cases, so that it causes erroneous operation. Therefore, VoIP service administrators are required to define secure rules for their own SIP headers; otherwise, the headers can be security holes.

7.2. SIP flooding attack detection results

The most significant consideration in SIP flooding detection is how to decide on a threshold that does not disturb VoIP services. To obtain the proper threshold, we monitored SIP messages between a UAC of a SIP network during the call-setup process and differentiated the messages according to state. Fig. 10 illustrates the number of transmitted SIP messages for each VoIP service. It shows that all five VoIP services transmitted SIP messages under 6 pps (packets per second) and 4 pps for the proceeding and completed states, respectively.

Based on the number of transmitted SIP messages under the normal VoIP service condition, we could now determine the state-based threshold for the message flooding attack. Assuming that there is no packet loss or retransmissions, we set the threshold for each state 2 pps higher than the highest pps for each state to

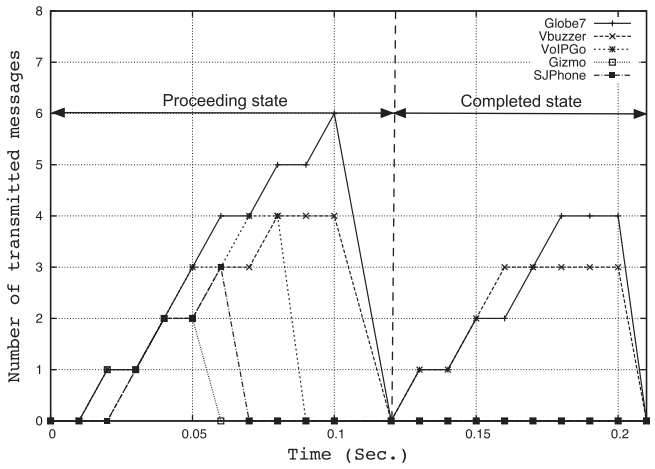

Fig. 10. The number of SIP messages for each state does not exceed 6 pps, and thus, we can differentiate the thresholds for each state.

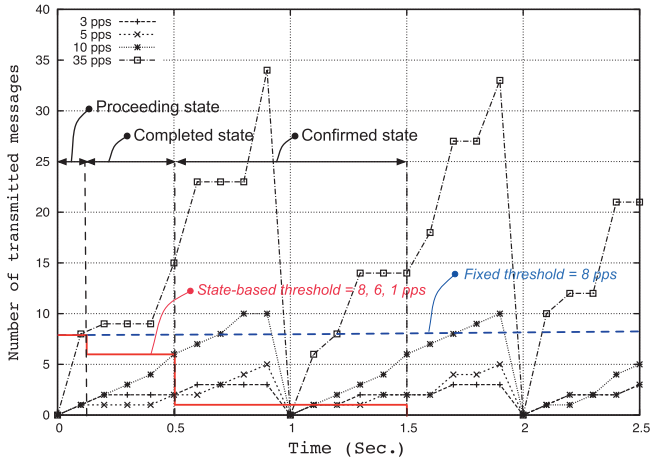

Fig. 11. To detect SIP message flooding, SIPAD uses adaptive thresholds (step-like solid line) as changing the current state while existing approaches uses a fixed threshold (horizontal dotted line). As the result, SIPAD is capable of detecting slow rate flooding such as 3 pps flooding.

reduce false alarms. In our experimental environment, a small VoIP network between a UAC and a UAS, the proper thresholds for the proceeding and completed states were 8 pps and 6 pps, respectively. For the confirmed state, 1 pps was set because the confirmed state only receives retransmitted ACKs and shortly transits to the terminated state.

To simulate the message flooding attack, we adjusted the SIP message generation rate from 1 pps to 35 pps. In fact, over 50 pps for VoIP services is easily suspected by simple packet monitoring, and 1 pps often occurs under normal conditions. Moreover, 1 pps is not a great burden for current computer systems, but anything greater than 3 pps starts consuming computer resources. Therefore, we applied four different generation rates using SiVuS: 3 pps, 5 pps, 10 pps and 35 pps.

Fig. 11 shows the message flooding simulation and the state-based threshold for each state. Existing flooding detection schemes employ a single fixed threshold (which is 8 pps in our experiment) and detect 35 pps and 10 pps flooding attacks at 0.12 s and 0.7 s, respectively. The fixed threshold cannot even detect 5 pps and 3 pps attacks since they never get to the 8 pps threshold. However, SIPAD varied its threshold from 8 pps to 1 pps as the state changed from the proceeding state to the confirmed state. As a result, SIPAD detected all the 30 pps, 10 pps, 5 pps, and 3 pps flooding attacks at

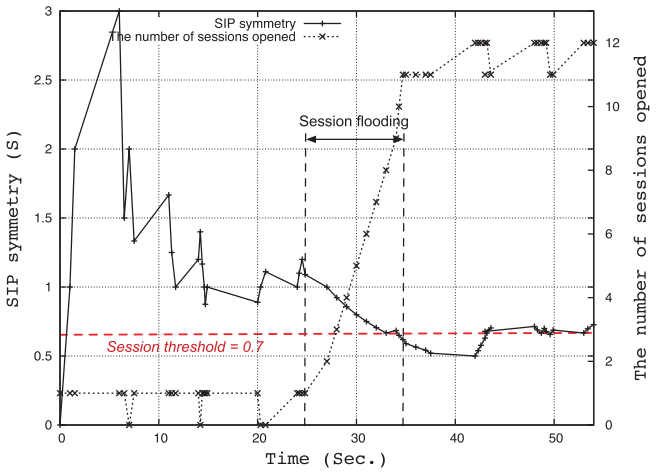

Fig. 12. To detect session flooding, SIPAD adtops the SIP message symmetry. In the experiment, SIP symmetry (S) is decreased under the symmetry threshold (0.7) as the session flooding attack is launched. After stopping the attack, S is eventually recovered.

0.12 s, 0.5 s, 0.5 s, and 0.5 s, respectively. By differentiating the threshold according to the state, SIPAD shows faster detection and higher detection accuracy by catching slow rate flooding attacks.

For session flooding detection, we monitored S based on the normal SIP–VoIP services that we aforementioned and found that S maintains higher than 0.8, and we set 0.7 as the symmetry threshold. Fig. 12 shows the result for the session flooding attack. Before the session flooding attack is launched, the caller and callee normally establishes and disconnects sessions, and S maintains higher value than the threshold[3]. During the slow rate session flooding attack with 1 pps from 25 s to 35 s, the callee receives relatively greater number of INVITE messages than the number of responses that he should send. In addition, the number of sessions is remarkably increasing because many different attackers flood INVITE messages. Therefore, S is rapidly decreased due to the unbalance between requests and responses, and then SIPAD can detect the session flooding attack. After finishing the attack, S eventually increases and will recover the normal value.

Of course, there is a possibility for larger VoIP networks to transmit normal SIP messages that are above the threshold. To adopt to different environments, a threshold based on machine learning is necessary but the principle used by SIPAD is still useful.

7.3. SIPAD overhead estimation

The major benefit of the stateful rule tree in SIPAD is to detect more SIP anomalies by rule hierarchy, as shown in Section 6.1. Another benefit is low overhead in terms of memory consumption and rule search efficiency. Here, we estimate the overhead under two different environments: the PC environment and the smartphone environment.

7.3.1. Memory consumption

- SIPAD in the PC: The memory requirement is approximately 11 MB, and most of which is used for Graphic User Interface (GUI) such as dialog and window controls. The SIPAD core algorithm only requires 2–3 MB. Moreover, it takes only 15 ms and 352 KB to load the rules. It is lightweight enough to be utilized in modern PCs.

[3] Note that the S was calculated on the callee's system so that the number of received messages are greater than the number of sent messages.

6-
IPAD in the smartphone: We implemented the SIPAD core algorithm in an existing open source Android VoIP application, called Sipdroid [30]. The size of Sipdroid increases by 20 KB (from 1.31 MB to 1.33 MB) after SIPAD was embedded in it, and the rules took 268 ms to load. Thus, SIPAD does not degrade Sipdroid performance in terms of call quality and application response time.

7.3.2. Rule search time

In Section 6.2, we insisted that SIPAD-SRT shows remarkably faster rule search time than sequential and binary searches. To verify the analysis, we implemented three different versions of SIPAD: SIPAD-Sequential, SIPAD-Binary, SIPAD-SRT (stateful rule tree). To see the processing time to find a rule depending on environments, each version was also implemented based on two environments: the PC environment and the smartphone environment. We then measured the rule search count and time taken for session establishment in the exchange of five SIP messages: INVITE, 100 Trying, 180 Ringing, 200 OK, and ACK. Table 5 depicts the comparison result. Note that the search time is based on our experimental environments, and the result can vary depending on processing power.

- Search count: SIPAD-SRT traversed the node 36.7 and 7.6 times less than SIPAD-Sequential and SIPAD-Binary, respectively.
- Search time in the PC: SIPAD-SRT found a rule 43.6 and 18.4 times faster than SIPAD-Sequential and SIPAD-Binary, respectively.
- Search time in the smartphone: SIPAD-SRT found a rule 12.1 and 7.2 times faster than SIPAD-Sequential and SIPAD-Binary, respectively.

These results are different from the analysis given in Section 6.2, which was that the binary search is 40 times faster than the sequential search. According to our experiments, however, the binary search is only 2 times faster than the sequential search. The reason is that the binary tree (alphabetically sorted by the rule's name) does not form a well-balanced binary tree; therefore, it cannot show ideal performance.

Since embedded devices such as smartphones or SIP hard-phones have limited resources (low processing power[4] and limited battery capacity), processing overhead (e.g., finding a rule from database) is a critical issue. Especially, the rule search count means the amount of operations performed and directly affects to the search time. That is, these two factors influences to power consumption for mobile devices. As SIPAD-SRT greatly reduces processing overhead with regard to the rule search count and time. As shown in Table 5, SIPAD-SRT only takes 15.85 ms to inspect SIP messages during session establishment, and it is negligible in the modern smartphone.

Moreover, VoIP network devices such as registrars and proxy servers need to process many SIP messages for call-setup and tear-down procedures. Assume that a registrar simultaneously handles 100 users who attempt to establish call sessions. Inspecting SIP messages from 100 sessions takes about 100 s (viz., Table 5) when using existing rule matching schemes. On the other hand, SIPAD-SRT takes only 2 s. SIP flooding attacks can amplify the effectiveness of the stateful rule tree, because the network devices receive numerous SIP messages that need to be inspected.

Based on the memory consumption and rule search time, SIPAD-SRT consumes reasonable amount of resources and is thus well suited for implementation in PCs and smartphones. In addition, it has a faster rule search efficiency than the other search methods.

[4] A VoIP hard-phone equips a VoIP processor that core is around 150–400 MHz [32,33].

Table 5
SIPAD-SRT searches the rules 7–43 times faster than the other rule search methods, depending on platform environments.

Search methods	Search count	Search time	
		PC (ms)	Smartphone (ms)
SIPAD-Sequential	1,008,302	108.78	192.04
SIPAD-Binary	210,661	45.91	114.41
SIPAD-SRT	27,426	2.49	15.85

Table 6
SIPAD detects more SIP attacks with higher accuracy than existing approaches. ∞ denotes non-detectable cases that cannot be detected within measurable time.

Approaches	Ehlert et al. [17]	Lahmadi et al. [19]	Seo et al. [31]	SIPAD
Malformed message detection (%)	74	74	100	100
Flooding detection (s)	0.7–∞	0.7–∞	0.7–∞	0.5–0.12
Slow rate flooding detection	N/A	N/A	N/A	O
Improper message transmission	O	O	O	O
Invalid header field	N/A	N/A	O	O
Invalid sub-rule	N/A	N/A	O	O
Search count	210–1008 k			27 k
Search time (ms)	45.91–192.04			2.49–15.85

7.4. Comparison with existing approaches

We compared SIPAD with three different SIP detection approaches. First, Ehlert et al. [17] pursued the similar goal that is to simultaneously detect malformed messages and flooding attacks. Second, Lahmadi et al. [19] developed a SIP-specified firewall, and it also utilizes state information. Last, Seo et al. [31] presented another approach to detect the two SIP attacks based on a rule matching algorithm.

Table 6 summarizes the experimental results comparison of SIPAD with existing approaches. Ehlert et al., Lahmadi et al. and Seo et al. can detect both malformed message and SIP flooding attacks, but they cannot detect slow rate flooding and message structural anomalies such as invalid header fields. Conversely, SIPAD builds a SIP-optimized structure and detects more SIP attacks with higher accuracy and lower system overhead. Thus, the existing approaches show partial detection for SIP attacks, while SIPAD achieves complete detection and fastest search time.

8. Related work

State machines have been utilized in intrusion detection by defining legitimate or abnormal cases. One such case is the State Transition Analysis Technique (STAT) [34], which is a rule-based intrusion detection approach. STAT is a general method that recognizes computer penetrations easily using a rule-based state diagram. There are different versions of STAT. NetSTAT [35] is used to determine which network event should be monitored, and WebSTAT [36] is to detect malicious behaviors targeted at web servers by analyzing web requests.

Snort [37] is the most widely deployed IDS around the world and has many attack patterns, over 6000. To protect VoIP systems, it may be possible to utilize an existing IDS. However, several challenges exist in the direct utilization of current IDSs to protect VoIP systems [38]. Firstly, VoIP service is based on session while IDS detects attacks based on packets. This means that an IDS monitors every single packet and compares it with pre-defined rules, but it

is necessary for a VoIP service to distinguish which session the packet belongs to. Secondly, although Snort provides stateful detection for TCP-based protocols such as HTTP and FTP, it does not help in processing stateful VoIP sessions. Lastly, a VoIP service utilizes a combination of protocols, such as the signaling protocol SIP and the media protocol RTP. If an attack is performed across protocols, conventional IDSs will fail to detect it. Therefore, it is necessary to develop intrusion detection technologies that are dedicated to VoIP services. A recent study [16] proposed VoIP IDS using statistical differences between VoIP Denial-of-Service (DoS) and flash crowds. It monitors state transitions and categorizes malicious behaviors according to three levels (transaction, sender, and global).

Several studies have been done regarding the protection of VoIP services. SCIDIVE [38] is an architecture that provides stateful and cross protocol detection. It is able to detect attacks in both the SIP and RTP protocols. To examine the SIP format, SCIDIVE uses rule sets including standard SIP rules. However, there are many malformed SIP messages that are formed as standard but are still dangerous. For example, `%s%d%caaa.com` follows a standard form, even though it may be dangerous because of the format string `%s%d`.

Sengar et al. [39] also proposed a VoIP defense mechanism using state machines. The mechanism uses cross protocol state machines that define attack detection patterns. The mechanism also has an advantage in that it detects across two protocols. However, it is not a flexible mechanism because it needs many state machines for protection against various attacks.

Geneiatakis et al. [13] proposed a framework to detect malformed SIP messages. Their proposal includes a framework based on the rules for valid SIP messages. The key idea is that normal SIP messages should have mandatory fields and conform to a pre-defined byte size. This mechanism, however, cannot detect elaborative malformed messages including mandatory and non-allowed fields. The non-allowed fields can cause unexpected results such as system crash. Considering the vulnerability, SIPAD inspects SIP messages using stateful information; therefore, we can detect not only malformed messages but also flooding attacks by the stateful inspection.

Chen [14] proposed a DoS detection method on SIP systems. It also utilizes RFC 3261 state transition models, and defines additional state and upper bounds for error conditions. One drawback of this approach is that it does not apply different thresholds depending on states, and thus it is vulnerable to slow rate flooding attacks. Another drawback is that malformed SIP messages are not considered properly. Although this mechanism can detect message flooding attacks, malformed SIP messages are definitely hazardous because they cause the malfunction of a VoIP service. In contrast to this approach, SIPAD is able to detect both malformed SIP messages and slow rate flooding attacks at the same time.

9. Conclusion

In this paper, we proposed a novel SIP–VoIP Anomaly Detection (SIPAD) scheme that detects both malformed SIP messages and SIP flooding attacks using a stateful rule tree. Instead of simply combining rule comparison and pre-defined thresholds, SIPAD builds a SIP-optimized rule tree based on secure SIP rules. Since SIPAD automatically constrains the search space, it can quickly detect not only two most significant types of attacks (malformed SIP messages and SIP flooding) but also three additional SIP anomalies: improper message transmission, invalid header field, and invalid subrules. Furthermore, SIPAD shows a 26% higher detection accuracy for malformed SIP attacks and a rule search time that is between 7 and 43 times faster than existing approaches.

Although we experimented using SIPAD on the PC and smartphone, it is possible to utilize an IDS/IPS module to protect VoIP networks, given that the core engine (stateful rule tree) has shown high accuracy and low overheads.

Acknowledgements

This research was supported by the R&BD Support Center of Seoul Development Institute and the South Korean government (WR080951) and the Public welfare&Safety research program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012M3A2A1051118 2012051118). The preliminary version was presented in IFIP TC-11 23rd International Information Security Conference (SEC) [31].

References

[1] S. McGann, D. Sicker, An analysis of security threats and tools in SIP-based VoIP systems, in: Proceedings of the Second Workshop on Securing Voice over IP, Cyber Security Alliance, 2005.
[2] T.J. Walsh, D.R. Kuhn, Challenges in securing voice over IP, IEEE Security & Privacy 3 (3) (2005) 44–49.
[3] P.C.K. Hung, M.V. Martin, Security issues in VoIP applications, in: CCECE, 2006, pp. 2361–2364.
[4] I. Packetizer, H.323 versus SIP: A Comparison, 2010. Available from: <http://www.packetizer.com/ipmc/h323_vs_sip>.
[5] Mobile voice & video calling: strategic opportunities & business models 2011–2016, Tech. Rep., Juniper Research, 2011.
[6] D. Geneiatakis, T. Dagiuklas, G. Kambourakis, C. Lambrinoudakis, S. Gritzalis, S. Ehlert, D. Sisalem, Survey of security vulnerabilities in session initiation protocol, IEEE Communications Surveys and Tutorials 8 (1–4) (2006) 68–81.
[7] D. Geneiatakis, C. Lambrinoudakis, An ontology description for SIP security flaws, Computer Communications 30 (6) (2007) 1367–1374.
[8] Common Vulnerabilities and Exposures (CVE), CVE via malformed SIP messages (2010–2012), 2012. Available from: <http://cve.mitre.org/>.
[9] C. Schanes, S. Taber, K. Popp, F. Fankhauser, T. Grechenig, Security test approach for automated detection of vulnerabilities of SIP-based VoIP softphones, International Journal On Advances in Security 4 (1–2) (2011) 95–105.
[10] Voice over Packet Security Forum, SiVuS: the VoIP Vulnerability Scanner, 2006. Available from: <http://www.vopsecurity.org/html/downloads.html>.
[11] Hewlett Packard (HP), SIPp, 2009. Available from: <http://sipp.sourceforge.net/index.html>.
[12] SX Design, Nastysip, 2004. Available from: <http://phoenix.labri.fr/documentation/sip/Documentation/Material/Clients/Tools/Test/NastySIP/SX%20Design.htm>.
[13] D. Geneiatakis, G. Kambourakis, T. Dagiuklas, C. Lambrinoudakis, S. Gritzalis, A framework for detecting malformed messages in SIP networks, in: Proceedings of Local and Metropolitan Area Networks (LANMAN), 2005.
[14] E. Chen, Detecting DoS attacks on SIP systems, in: Proceedings of VoIP Management and Security, 2006.
[15] H. Sengar, H. Wang, D. Wijesekera, S. Jajodia, Detecting VoIP floods using the hellinger distance, IEEE Transactions on Parallel Distributed Systems 19 (6) (2008) 794–805.
[16] S. Ehlert, Y. Rebahi, T. Magedanz, Intrusion detection system for Denial-of-Service flooding attacks in SIP communication networks, International Journal of Security and Networks (IJSN) 4 (3) (2009) 189–200.
[17] S. Ehlert, G. Zhang, D. Geneiatakis, G. Kambourakis, T. Dagiuklas, J. Markl, D. Sisalem, Two layer Denial of Service prevention on SIP VoIP infrastructures, Computer Communications 31 (10) (2008) 2443–2456.
[18] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Spark, M. Handley, E. Schooler, RFC3261: Session Initiation Protocol (SIP), 2002.
[19] A. Lahmadi, O. Festor, SecSip: a stateful firewall for SIP-based networks, in: Proceedings of the 11th IFIP/IEEE International Conference on Symposium on Integrated Network Management, 2009, pp. 172–179.
[20] M. Handley, V. Jacobson, RFC2327: Session description protocol (SDP), 1998.
[21] Y. Rebahi, D. Sisalem, T. Magedanz, SIP Spam Detection, in: Proceedings of International Conference on Digital Telecommunications (ICDT), 2006.
[22] A.D. Keromytis, A comprehensive survey of voice over IP security research, IEEE Communications Surveys and Tutorials 14 (2) (2012) 514–537.
[23] S. Vuong, Y. Bai, A survey of VoIP intrusions and intrusion detection systems, in: The 6th International Conference on Advanced Communication Technology (ICACT), 2004, pp. 317–322.
[24] S. Ehlert, D. Geneiatakis, T. Magedanz, Survey of network security systems to counter SIP-based denial-of-service attacks, Computers & Security 29 (2) (2010) 225–243.
[25] R. Zhang, X. Wang, R. Farley, X. Yang, X. Jiang, On the feasibility of launching the man-in-the-middle attacks on VoIP from remote attackers, in: ASIACCS, 2009, pp. 61–69.

[26] R. Zhang, X. Wang, X. Yang, X. Jiang, On the billing vulnerabilities of SIP-based VoIP systems, Computer Networks 54 (11) (2010) 1837–1847.
[27] D. Geneiatakis, N. Vrakas, C. Lambrinoudakis, Utilizing bloom filters for detecting flooding attacks against SIP based services, Computers & Security 28 (7) (2009) 578–591.
[28] Computer Engineering Laboratory, University of Oulu, PROTOS Test-Suite:c07-sip, 2005. Available from: <http://www.ee.oulu.fi/research/ouspg/protos/testing/c07/sip/index.html>.
[29] MyVoIPProvider.com, Top 100 VoIP Providers World Ranking, 2012. Available from: <http://www.myvoipprovider.com/>.
[30] Sipdroid, Free SIP/VoIP client for Android, 2012. Available from: <http://sipdroid.org/>.
[31] D. Seo, H. Lee, E. Nuwere, Detecting more SIP attacks on VoIP services by combining rule matching and state transition models, in: IFIP TC-11 23rd International Information Security Conference (SEC), 2008, pp. 397–411.
[32] Analog Devices, Selecting a Processor for VoIP Solutions, 2008.
[33] Broadcom, BCM1161 Mobile Processor, 2012. Available from: <http://ko.broadcom.com/products/Wireless-LAN/Wi-Fi-Phone-Solutions/BCM1161>.
[34] K. Ilgun, R.A. Kemmerer, P.A. Porras, State transition analysis: a rule-based intrusion detection approach, IEEE Transactions on Software Engineering 21 (1995) 181–199.
[35] G. Vigna, R. Kemmerer, NetSTAT: A network-based intrusion detection approach, in: Proceedings of the 14th Annual Computer Security Application Conference (ACSAC), 1998.
[36] G. Vigna, W. Robertson, V. Kher, R. Kemmerer, A stateful intrusion detection system for world-wide web servers, in: Proceedings of the Annual Computer Security Applications Conference (ACSAC), 2003.
[37] M. Roesch, Snort-lightweight intrusion detection for networks, in: Proceedings of USENIX LISA, 1999.
[38] Y.-S. Wu, S. Bagchi, S. Garg, N. Singh, SCIDIVE: A stateful and cross protocol intrusion detection architecture for Voice-over-IP environments, in: Proceedings of International Conference on Dependable Systems and Networks (DSN), 2004.
[39] H. Sengar, D. Wijesekera, H. Wang, S. Jajodia, VoIP intrusion detection through interacting protocol state machines, in: Proceedings of International Conference on Dependable Systems and Networks (DSN), 2006.