Hindawi
Security and Communication Networks
Volume 2020, Article ID 8853625, 14 pages
https://doi.org/10.1155/2020/8853625

Research Article

A Diagnosis and Hardening Platform for an Asterisk VoIP PBX

Pelayo Nuño, Carla Suárez, Eva Suárez, Francisco G. Bulnes, Francisco J. delaCalle, and Juan Carlos Granda

Department of Computing, University of Oviedo, Campus de Viesques, Gijón, 33204, Asturias, Spain

Correspondence should be addressed to Pelayo Nuño; nunopelayo@uniovi.es

Received 8 May 2020; Revised 26 November 2020; Accepted 4 December 2020; Published 29 December 2020

Academic Editor: Flavio Lombardi

Voice over IP (VoIP) is a set of software and hardware technologies used for making voice calls over the Internet. VoIP has been massively deployed in corporative environments since voice and data network convergence enables unified communication services while reducing costs. The main component of a VoIP network infrastructure is the private branch exchange (PBX). Nowadays, Asterisk is the most widespread PBX deployed within corporations due to its open access technology, along with its modular and flexible design. The configuration of PBX systems usually relies on multiple configuration files composed of a vast number of parameters that may have an impact on the security of the system. Therefore, the setup of such systems tends to be complicated and prone to errors and usually requires highly specialized human intervention. In this research, a diagnosis platform for discovering vulnerabilities and security breaches in the configuration of an Asterisk PBX is presented. The proposed platform performs both reactive and proactive actions in order to reconfigure and harden an Asterisk PBX. Firstly, the platform reacts after certain events by modifying the configuration of the Asterisk PBX in order to mitigate risks. Secondly, the platform performs several on-demand assessments that also reconfigure the Asterisk PBX to improve overall security. Finally, the functionality of the platform is easily extensible and highly customizable. Extensive tests have been carried out to assess the security and performance of the Asterisk PBX when facing attacks. Results show that the security of the platform increases, avoiding performance degradation when using the proposed platform.

1. Introduction

Voice over IP (VoIP) encompasses a set of software and hardware technologies to make voice calls using data networks as an alternative to the traditional public switched telephone network (PSTN) system. VoIP is widely used in corporative environments, and the adoption of this technology in businesses is expected to keep increasing in future years [1]. The main reason behind the popularity of this paradigm is cost saving. Both large and small companies have realized that deploying and managing separated data and voice networks is expensive. In contrast, voice and data network convergence enables unified communication services while reducing costs. Moreover, costs associated with traditional phone calls are usually more expensive than costs associated with VoIP calls [2].

One of the key components of a VoIP network is the private branch exchange (PBX). Within the scope of a corporate site, Asterisk has become the most convenient solution. Asterisk is an open source Linux-based PBX that is gaining momentum in the VoIP industry [3–6]. It is quite feasible to develop an inexpensive enriched IP telephony service with Asterisk [7], supported by standard and widely used protocols such as Session Initiation Protocol (SIP) and Real-time Transport Protocol (RTP). Asterisk allows communication between terminals with one or more lines connected to the PSTN. Therefore, terminals can establish calls over the Internet or with other terminals in the PSTN. Other functionalities provided by Asterisk are voicemail, call scheduling, and interactive voice response (IVR), to name a few.

VoIP networks are exposed to risks and attacks similar to those suffered by other Internet technologies. Denial of service (DoS) attacks, unpatched vulnerabilities exploitation, or malicious login attempts are examples of common attacks. Furthermore, VoIP must cope with challenges specific to its activity such as toll fraud, VoIP spam, phishing, and voice impersonation. Finally, as PBXs are programmable, these environments also pose new risks. Dial plan vulnerabilities like using deprecated or compromised methods or programming flaws which lay them open to dial plan injection scenarios are examples of risks. Therefore, security for Asterisk PBXs must be a key concern.

PBX systems like Asterisk have many configuration parameters that are directly or indirectly related to security. The setup and deployment of such systems requires human intervention, typically highly specialized staff. This may be a complex and daunting task. Usually, Asterisk setup involves dozens of configuration files and hundreds of source lines of code per file. Therefore, the manual management of these systems tends to be complicated and prone to errors. In fact, the exposure of the VoIP service strongly depends on configuration vulnerabilities [8].

In this work, a diagnosis platform for discovering vulnerabilities and security breaches in the configuration of an Asterisk PBX is presented. The proposed platform performs reactive actions to reconfigure and harden the Asterisk PBX. Specifically, the platform is able to perform, but not limited to, the following securing actions. Firstly, the platform reacts to account scans and malicious login attempts by updating access control rules. Secondly, the platform checks the correctness of the programmed dial plan and automatically fixes source code lines that compromise the security of the PBX. Thirdly, the platform automatically replaces deprecated and compromised methods and functions. Finally, the platform can be easily managed to include new vulnerabilities to evaluate in the assessment process in order to avoid premature obsolescence.

The remainder of the paper is organized as follows. Section 2 briefly describes the architecture and configuration of an Asterisk PBX. Related work about security in Asterisk is presented in Section 3. Security concerns of Asterisk are explained in Section 4. Section 5 details the architecture of the proposed platform and describes the diagnosis assessments and hardening actions that are carried out to secure an Asterisk PBX. Experimentation and evaluation of these assessments and actions are discussed in Sections 6 and 7. Finally, Section 8 contains the concluding remarks and outlines future work.

2. Background and Basic Concepts

A VoIP call is composed of two planes: call signalling and voice data. Asterisk behaves as a back-to-back user agent (B2BUA) in a VoIP call, operating between the participants both in the signalling and data planes as illustrated in Figure 1. Configuring Asterisk for data to flow directly between participants (pass-through mode) is also possible. In this case, Asterisk operates between participants only in the signalling plane, but voice data flow between participants directly. Asterisk supports several standard and proprietary protocols for call signalling such as SIP, H.323, Skinny Client Control Protocol (SCCP), and Inter-Asterisk eXchange (IAX).

Asterisk configuration relies on multiple configuration files. Each signalling protocol is related to one or several files,

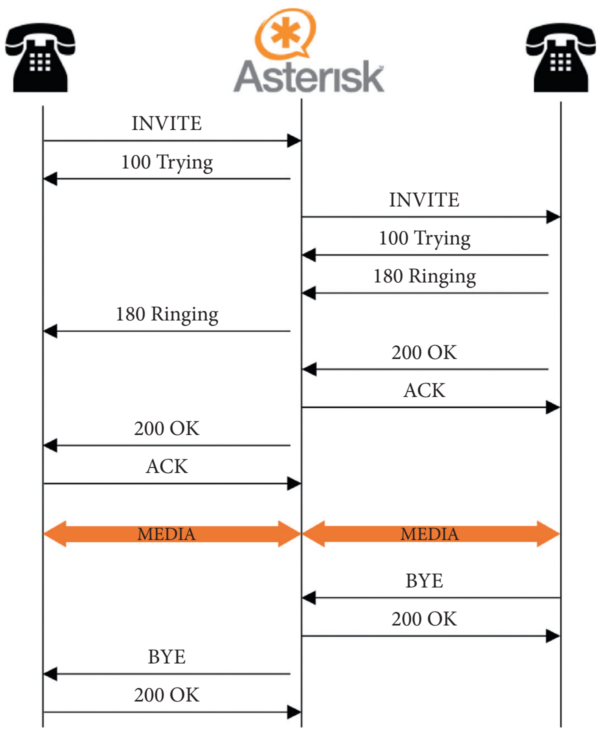


Figure 1: Asterisk B2BUA.

depending on the Asterisk version. Some configuration files of signalling protocols, e.g., *pjsip.conf* for SIP, declare the accounts that can login in the PBX by using such signalling protocol. Authentication parameters like usernames and passwords, supported codecs, or transport protocols can be defined. Asterisk functionality is also programmed by means of a special configuration file (*extensions.conf*) that defines the dial plan. This configuration file is the core of the PBX, where actions and services that are available for an account are specified in a declarative way.

Figure 2 shows configuration examples of the SIP signalling and the dial plan in Asterisk. A basic definition of an endpoint called Alice using SIP as the signalling protocol is illustrated in Figure 2(a). The configuration is organized in four sections. The first section is transport, where several types of transport protocols for exchanging SIP messages between terminals and Asterisk can be configured. Next, the end point section is used to configure SIP-related parameters. In the figure, the endpoint is configured to send data using pass-through mode when possible (directmedia = yes). Furthermore, the end point section is linked to other sections such as transport, authorization (auth), address of record (aors), and dial plan (context). Credentials related to the endpoint such as username and password are defined in the auth section. Finally, the aor section is used to instruct Asterisk about where Alice is located. This can be done statically, by indicating an IP address, or dynamically so that the endpoint must perform an SIP registration process in order to be dialled.

Figure 2(b) shows the actions that *Alice's* endpoint is able to perform through the Asterisk PBX. As can be seen, there are several ways of calling the same endpoint (Bob) by

```
[TypeTransport]
type=transport
protocol=udp
bind=0.0.0.0

[Alice]
type=endpoint
transport=TypeTransport
disallow=all
allow=alaw
allow=ulaw
aors=Alice
directmedia=yes
auth=AliceCredentials
context=DialplanAlice

[Alice]
type=aor
max_contacts=1

[AliceCredentials]
type=auth
auth_type=userpass
username=Alice
password=unsecure_password
```

(a)

```
[DialplanAlice]
exten => 1000,1,Dial(PJSIP/Bob)
;Bob's terminal is called if Alice dials the
extension Nos. 1000

exten => _100X,1,Dial(PJSIP/Bob)
;Bob's terminal is called if Alice dials an extension
between Nos. 1000 and 1009

exten => _100Z,1,Dial(PJSIP/Bob)
;Bob's terminal is called if Alice dials an extension
between Nos. 1001 and 1009

exten => _100N,1,Dial(PJSIP/Bob)
;Bob's terminal is called if Alice dials an extension
between Nos. 1002 and 1009

exten => _10[2-6]0,1,Dial(PJSIP/Bob)
;Bob's terminal is called if Alice dials any
extension from {1020, 1030, 1040, 1050, 1060}

exten => _10.,1,Dial(PJSIP/Bob)
;Bob's terminal is called if Alice dials 10 and one
or more characters

exten => _10!,1,Dial(PJSIP/Bob)
;Bob's terminal is called if Alice dials 10 and zero
or more characters
```

(b)

Figure 2: Asterisk configuration examples. (a) pjsip.conf file. (b) extensions.conf file.

dialling different extensions, some of them using extension patterns. Extension patterns can match digits (X, Z, and N), any digit or character within the interval denoted with square brackets, and wildcards. These wildcards are powerful tools that can match zero or more characters (!) or one or more characters (.) immediately after the given partial extension.

In addition to the configuration files, the functionality offered by Asterisk can be extended using two APIs. First, the Asterisk Gateway Interface (AGI) can be used to trigger other applications when events in current calls arise. A version of this API, called FastAGI, allows triggering applications in remote machines through TCP sockets. Second, the Asterisk Manager Interface (AMI) can be used to control Asterisk from other applications or services. This API can be used to initiate calls, configure voicemail, redirect calls, and so on.

The AMI interface allows an external system or agent to connect to an Asterisk PBX and execute commands, retrieve information, or capture events. Figure 3 shows the three type of messages exchanged using the AMI. These messages have the same syntax, which is composed of several fields, where each field is a key-value pair delimited by a colon. First, Figure 3(a) shows an action, which is sent by the connected agent when requesting the execution of a particular task in the PBX such as login remotely, getting a configuration file in order to be parsed, or adding a new extension in the dial plan. Second, Figure 3(b) corresponds to a response event, which is sent by Asterisk to reply the last action requested by the connected agent. As can be seen, every action and its subsequent response are paired with the ActionID field. Third, Figure 3(c) illustrates an operational event that is raised by Asterisk when a specific circumstance occurs, such as an invalid login attempt.

The AMI is a key component of the architecture presented in this work, and its purpose is twofold. First, events from a predefined set are retrieved from the Asterisk PBX in order to perform diagnosis. Second, reconfiguring actions to modify the Asterisk dial plan are executed through this API.

3. Related Work

Security in VoIP networks is a wide research topic that is not focused on a single standard but on several technologies, entities, and multiple vectors of attack [9]. For example, an analysis of the vulnerabilities to cope with in an H.323-based IP telephony scenario composed of several PBXs and gateways is examined in [10]. The authors describe gatekeeper registration and gatekeeper DoS as the two main attacks to the VoIP infrastructure targeting the PBX.

Nowadays, SIP is the major signalling protocol in VoIP networks, so there are many related research works. A comprehensive classification of SIP attacks according to multiple aspects, such as origin and consequences, can be found in [11], while mechanisms to mitigate them are surveyed in [12, 13]. Some research works are focused on enhanced SIP security in Asterisk. A call hijacking attack compromising the Asterisk implementation of SIP is discovered in [14]. Both authenticity and no repudiation of calls are jeopardized. In [15], a SIP-based secured architecture using cryptographic token for authentication (following the concept of Single Sign-On) is designed and developed. This solution implies adding a new security layer to the SIP standard. The authors extend the capabilities of an Asterisk

```
Action: DialplanExtensionAdd
ActionID: 1234
Context: default_context
Extension: default_extension
Priority: 1
Application: Answer
```

(a)

```
Response: Success
ActionID: 1234
Message: Added requested extension
```

(b)

```
Event: InvalidPassword
Privilege: security,all
EventTV: 2020-10-13T17:03:59.153+0000
Severity: Error
Service: SIP
EventVersion: 2
AccountID: 100
SessionID: ...
LocalAddress: ...
RemoteAddress: ...
Challenge: ...
ReceivedChallenge: ...
ReceivedHash: ...
```

(c)

Figure 3: Type of messages using the AMI. (a) Action. (b) Response. (c) Event.

PBX in order to be interoperable with the proposed SIP implementation.

Recently, several research works have been published aiming to improve the security of an Asterisk PBX. In [16], a defense system against distributed DoS (DDoS) attacks based on captchas for programmable PBXs is proposed. These captchas can only be solved by human users, so calls from zombie VoIP phones are discarded and only legitimated users are allowed to interact with the PBX. An approach for avoiding Spam over IP Telephony (SPIT) in Asterisk environments is developed in [17]. They implement a caller blacklist that is updated in real time according to data extracted from Call Detail Records (CDRs). Spammers are banned for a length of time that is gradually incremented after replays. However, using Asterisk ACLs or Linux iptables to further block the IP address related to spammers is not considered.

In [18], an intrusion prevention middleware is introduced. Caller and call information are detached from Asterisk, stored in an independent relational database, and data marshalling and demarshalling is used to prevent information theft. A secure platform for mobile VoIP services based on Asterisk is presented in [19]. The proposed secure mobile VoIP platform brings protection against eavesdropping and DoS attacks using secure voice coding and real-time migration. In addition, an architecture based on a honeypot to identify attacks and inform Asterisk is described in [20]. This platform detects SIP port scanning, VoIP service abuse attack, and VoIP MAC spoofing. Nevertheless, Asterisk PBX is not dynamically reconfigured.

An online risk management platform for enterprise VoIP is implemented into Asterisk in [8]. This platform dynamically adapts the exposure of the PBX through the application of real-time countermeasures in order to mitigate security attacks. A risk model is defined, and the network is continuously assessed against this model in order to evaluate the network exposure. The platform can detect and react when facing unauthorized login attempts, SPIT calls, and eavesdropping. If a potential threat is detected, the platform dynamically activates security countermeasures such as redirecting, hanging up, or putting the call on hold. It is built on four entities: a call monitor for collecting data from VoIP calls, a threat detector for identifying potential attacks, a risk manager for assessing threats against the model, and finally, a configuration manager for conveying countermeasures to VoIP devices in the network.

As suggested by the authors in [8], the automatic identification of configuration vulnerabilities may improve the assessment of the VoIP network exposure and the risk management performance. Thus, this paper proposes a platform including a proactive strategy for discovering vulnerabilities. Not only can the proposed platform detect and react to some types of attacks in real time, but it also proactively evaluates the setup of the Asterisk PBX by parsing its configuration files. This allows for detecting threats resulting from programming bugs and risky coding habits both in the dial plan and the configuration files of the Asterisk PBX. Furthermore, these threats are mitigated by automatically fixing such issues, so vulnerabilities that would leave the door open for potential attacks to the VoIP network are corrected beforehand.

4. Security Concerns in Asterisk

In [21], VoIP security threats are organized in four main categories: attacks aiming at disrupting VoIP service availability and malicious activities attempting to compromise integrity of services such as toll fraud, SPIT, and eavesdropping. Security concerns are further classified as external or internal threats [22]. External threats arise from individuals without authorized access to the computer systems or network. Within the context of this work, they are threats that compromise the configuration of signalling protocols or the Asterisk PBX as a whole. On the other hand, a threat is considered internal to the organization as the result of an employee action or failure of an organization process, e.g., the existence of programming flaws. Thus, this category is strictly related to the Asterisk dial plan configuration.

4.1. External Threats. When deploying a publicly exposed PBX, there are several external threats to cope with, independently of the signalling protocol involved in the communications. First, accounts are scanned in an attempt to guess legitimate usernames and passwords. Second, a variant of this approach when the flooding of scans is oriented causes a DoS in the PBX. Third, if a relational database is used to validate the credentials of the login attempts, such database can be exploited through SQL injection attacks [11].

The aforementioned threats can be observed in Figure 4, where the log of an Asterisk PBX deployed in Microsoft Azure with a public IP address is shown, but IP addresses of the attackers are anonymized. As can be seen, account scans are continuously received. Notices with numbers 35728, 35939, and 36106 are examples of password guessing. These scans are checking common usernames used for registering endpoints in the PBX within corporative environments. This is just a first step, since the goal is to determine whether the account exists based on the response.

If the PBX is misconfigured, instead of sending a generic denied access message, a response will be sent indicating whether the account exists or not. If the account exists, the PBX will request authentication while a deny registration response will be received if the account does not exist. Therefore, in the first case, a valid username would be discovered, so both brute force and dictionary attacks could be performed in order to find out the password related to such a user.

The remaining entries are examples of SQL injection attacks where malformed usernames are crafted in an attempt to compromise the PBX.

4.2. Internal Threats. The external threats can be mitigated to a limited degree with a proper configuration of the signalling channels in the PBX. For example, if the channel used for processing SIP requests is the SIP channel (*sip.conf*), the option `alwaysauthreject` must be enabled. With such a configuration, rejected incoming SIP requests get an identical response, as if the username was actually registered, making it impossible for the attackers to guess valid usernames through an account scan. Similarly, if the channel used for signalling is the IAX channel (*iax.conf*), the option `delayreject` must be activated. Thus, Asterisk delays sending the rejected response in subsequent authentication attempts, which increases the exposure time needed by brutal force attacks to succeed.

Furthermore, some signalling channels have options enabled by default that can pose risks. This is the case of the option `allowguest` in channel *sip.conf*, which permits unregistered callers to call the PBX. This situation can be even more dangerous if these anonymous callers do not have their own and isolated context configured in the dial plan, since they would be able to perform the same actions as a legitimated user.

However, the main internal threat related to Asterisk is a badly configured dial plan. Specifically, the most dangerous vulnerability is the concept of dial plan injection. An example of dial plan allowing for exploiting this vulnerability is shown in Figure 5, in which the dial plan consists of just one single extension, labelled *A*. This extension has an extension pattern ending with a period character, which matches one or more characters after a first digit between 0 and 9 (*X*). Therefore, this extension pattern processes every single dialled extension that begins with a digit and subsequently calls the extension that is stored in the Asterisk system variable `${EXTEN}`, using SIP as a signalling protocol through the PJSIP channel.

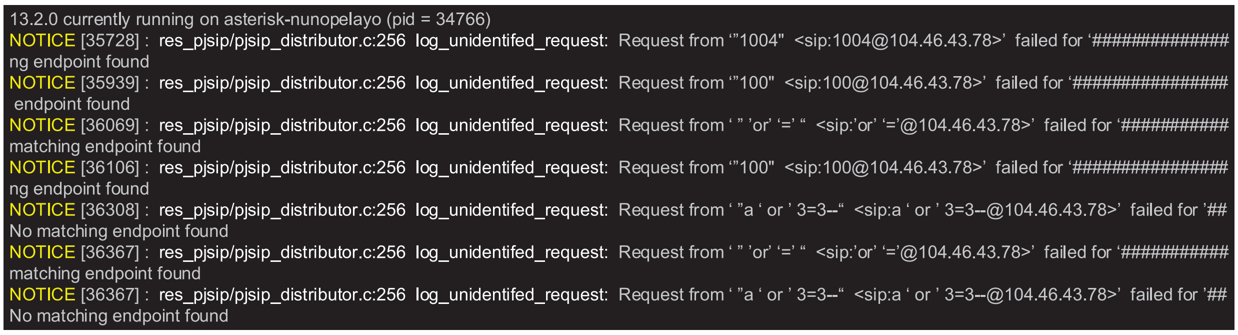
The problem arises if an extension which is both malicious an alphanumeric is dialled, like the one shown in *B*. Supposing that *Alice* has extension 2500 assigned, *Trudy* dials the aforementioned malicious extension, *B*, which will be interpreted, as represented in *C*, as a call to two destinations. The first one uses the PJSIP channel to call extension 2500, which belongs to Alice. The second one will try to use the Asterisk DAHDI channel to call number 900123456 through the PSTN, so it could be a deliberate high-cost call.

In addition, the dial plan can pose various threats if the source code is using deprecated or unpatched Asterisk functions, which expose them to attacks on well-known vulnerabilities. There are several Asterisk functions, in different versions, that may compromise the confidentiality, integrity, and availability of the PBX, such as the following examples. CVE-2012-2414 vulnerability, which is related to the `SHELL` function, and CVE-2017-7617, which corresponds to the `CDR` function, may lead to remote code execution. Similarly, CVE-2014-8417 and CVE-2014-8417, related to `CONFBRIDGE` and `DB` functions, respectively, may allow remote authenticated users to gain system privileges. Finally, CVE-2014-6610 is a vulnerability of the `RECEIVEFAX` function that can cause a DoS in the Asterisk PBX.

To conclude, inadequate Asterisk hardening may increment security risks. The transport protocols for signalling traffic must be configured with care, avoiding binding more interfaces than strictly necessary for listening to SIP traffic and forcing the use of secure protocols such as Transport Layer Security (TLS) whenever possible. It is also recommended not to use default ports for signalling protocols as part of security best practices. Moreover, Asterisk provides Access Control Lists (ACLs) to perform packet filtering on the ingress traffic. However, there is an implicit *permit any* rule by default at the end of every ACL, so anything that is not explicitly denied is permitted. Therefore, it is highly advisable to configure ACLs starting with a *deny any* rule and, after that, several permit rules for the desired networks. Finally, like in other environments, passwords must be chosen according to strict policies to avoid password guessing attacks where dictionary brute-force methods against accounts with weak passwords are carried out.

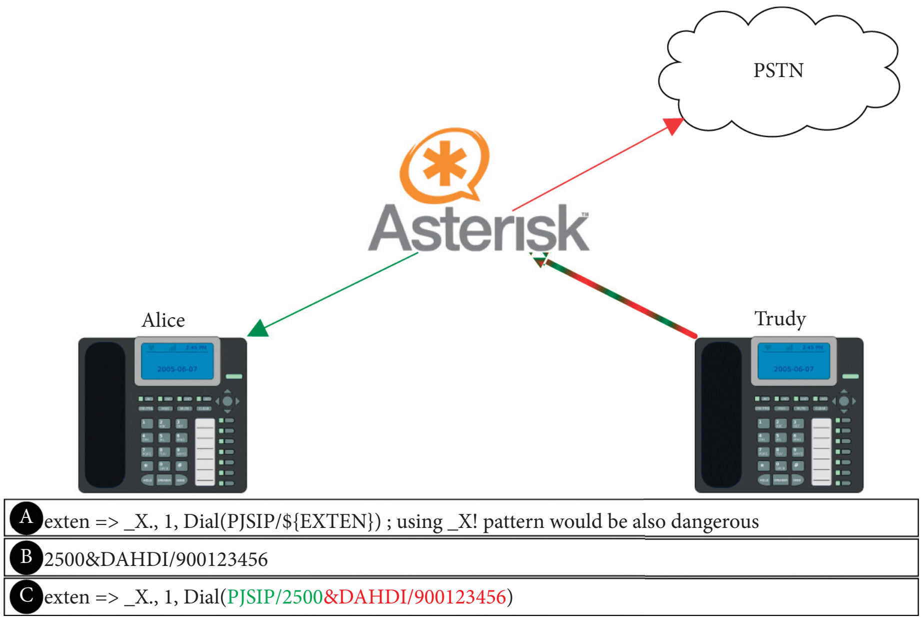
5. Proposed Platform

The proposed platform follows both reactive and proactive approaches. The platform performs reactively when threatening situations are detected, and, as a result, the configuration of the Asterisk PBX is modified in real time. Furthermore, it follows a proactive approach in order to diagnose and harden the Asterisk PBX. As stated in [21], proactive strategies identify and mitigate specific vulnerabilities related to VoIP before they affect end users. Therefore, once identified, security vulnerabilities should be addressed by appropriate actions, such as reconfiguring or



```
13.2.0 currently running on asterisk-nunopelayo (pid = 34766)
NOTICE [35728] :  res_pjsip/pjsip_distributor.c:256  log_unidentifed_request:  Request from '"1004"  <sip:1004@104.46.43.78>'  failed for '##############
ng endpoint found
NOTICE [35939] :  res_pjsip/pjsip_distributor.c:256  log_unidentifed_request:  Request from '"100"  <sip:100@104.46.43.78>'  failed for '################
 endpoint found
NOTICE [36069] :  res_pjsip/pjsip_distributor.c:256  log_unidentifed_request:  Request from ' " 'or' '=' "  <sip:'or' '='@104.46.43.78>'  failed for '###########
matching endpoint found
NOTICE [36106] :  res_pjsip/pjsip_distributor.c:256  log_unidentifed_request:  Request from '"100"  <sip:100@104.46.43.78>'  failed for '################
ng endpoint found
NOTICE [36308] :  res_pjsip/pjsip_distributor.c:256  log_unidentifed_request:  Request from ' "a ' or ' 3=3--"  <sip:a ' or ' 3=3--@104.46.43.78>'  failed for '##
No matching endpoint found
NOTICE [36367] :  res_pjsip/pjsip_distributor.c:256  log_unidentifed_request:  Request from ' " 'or' '=' "  <sip:'or' '='@104.46.43.78>'  failed for '###########
matching endpoint found
NOTICE [36367] :  res_pjsip/pjsip_distributor.c:256  log_unidentifed_request:  Request from ' "a ' or ' 3=3--"  <sip:a ' or ' 3=3--@104.46.43.78>'  failed for '##
No matching endpoint found
```

Figure 4: Account scan attacks over an Asterisk PBX deployed in Microsoft Azure.



A exten => _X., 1, Dial(PJSIP/${EXTEN}) ; using _X! pattern would be also dangerous

B 2500&DAHDI/900123456

C exten => _X., 1, Dial(PJSIP/2500&DAHDI/900123456)

Figure 5: Dial plan injection attack from Trudy.

patching, in order to increase the hardening of the Asterisk PBX.

5.1. Architecture of the Platform. The architecture of the diagnosis and hardening platform is illustrated in Figure 6. As can be seen, it is composed of four entities with different roles. An Asterisk PBX configured to send operational events through the AMI interface is deployed. These events are captured by a special entity called the receiver. The receiver exclusively collects security events, storing them in a relational database for further processing. Table 1 shows the set of events related to security that can be raised from an Asterisk PBX [23]. The receiver is the only entity with privileges to receive events from the PBX by using the AMI. Although it is detached from the controller in the figure for the sake of clarity, both entities could be located in the same computer.

The controller reacts after every incoming security event that is collected and stored in the database. The controller processes the security events and is able to autonomously update the configuration of the Asterisk PBX. For example, after receiving a configurable number of unauthorized login attempts within an also configurable interval of time, the controller inserts a new access control rule in the PBX to deny the access from the network where the source IP address is located.

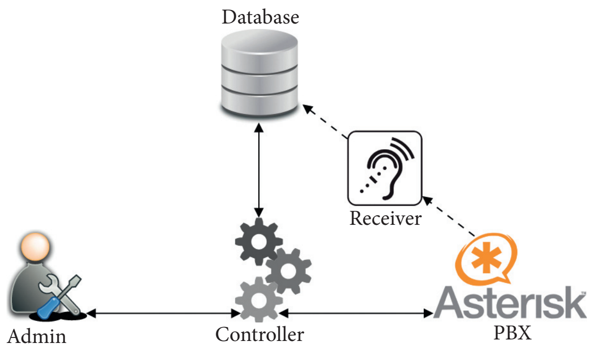


Figure 6: Architecture of the platform.

The database triggers a notification after each event is inserted by the receiver. This notification, which is handled by the controller, conveys information about the type of

Table 1: Security events raised by the PBX through the AMI.

Event name	Situation when it is raised
AuthMethodNotAllowed	A request used an authentication method not allowed by the service
ChallengeSent	An Asterisk service sends an authentication challenge to a request
FailedACL	A request violates an ACL check
FullyBooted	All Asterisk initialization procedures have finished
InvalidAccountID	A request fails an authentication check due to an invalid account ID
InvalidPassword	A request provides an invalid password during an authentication attempt
LoadAverageLimit	A request fails because a configured average load limit has been reached
MemoryLimit	A request fails due to an internal memory allocation failure
RequestBadFormat	A request is received with bad formatting
RequestNotAllowed	A request is not allowed by the service
RequestNotSupported	A request fails due to some aspect of the request not being supported by the service
SessionLimit	A request fails due to exceeding the number of allowed concurrent sessions for the service
SuccessfulAuth	A request successfully authenticates with a service
UnexpectedAddress	A request has a different source address from the expected for an ongoing session

event, a timestamp, and the source IP address of the suspect. If the event is related to a failed login attempt, the controller checks if the suspect has exhausted the maximum number of login attempts in the configured time interval. To do so, it uses a moving window to keep in memory all the events related to unauthorized login attempts that have been processed during this interval. The main advantage of this design compared to periodic polling is a shorter reaction time when facing a potentially dangerous event. However, this also implies more overhead on the controller when the number of concurrent events is extremely high.

Trustworthy and well-known networks or host IP addresses can be excluded from this process by creating a white list, preventing legitimate users from being blocked due to an erroneous account setup in their VoIP devices. The controller is the only entity with privileges to modify the configuration of the PBX by using the AMI. Updates performed by the controller modify the configuration files of the Asterisk PBX, so these changes are persistent. Furthermore, to face especially crucial events, the AMI allows reloading the configuration files in order to apply changes immediately.

The platform monitors the events through a web control panel. They are classified according to both the potential danger and the reason that fired the event in the PBX. A subsequent classification is also performed in order to distinguish events whose causes have been solved from those events with pending resolution.

5.2. Diagnosis Assessments. The controller can also be instructed by the administrator, via the web control panel, to perform several on-demand diagnosis and hardening assessments with different purposes. These assessments are carried out through the AMI. As a result, the configuration of the Asterisk PBX can be modified if vulnerabilities or misconfigurations are found.

One of the goals of the proposed platform is to make sure the dial plan used by the PBX is secure. This implies carrying out two types of assessment. The first is to verify that dial plan injection attacks cannot be conducted. The second is to check that the dial plan is not using any function considered deprecated or potentially unsafe. To do so, a parser to process the *extensions.conf* file is built. This parser, created using *JFlex* and Yacc, in addition to verifying that the *extensions.conf* file is lexically and syntactically correct, looks for extensions that are susceptible to dial plan injection attacks and lines of code where unsafe functions are used. An excerpt of the grammar used to build the parser, expressed in BNF notation, is shown in Figure 7.

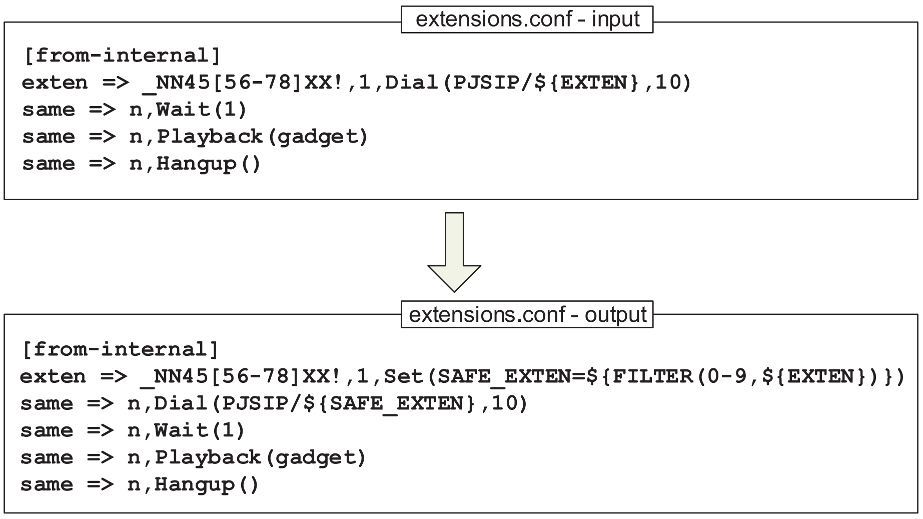
The code executed by the parser to reduce most of the rules is simple: it copies the lexemes of the tokens processed on a list, which will be later dumped into an output file. Thus, if the dial plan is correct and secure, the input and output will be identical. However, as stated in Section 4.2, if an extension susceptible to dial plan injection attacks is detected, the dial plan will be modified automatically to prevent these attacks. When the parser detects an extension ending with a wildcard calling function *Dial*, the dial plan will be modified. Function *Dial* will be called using a filter that only allows the user to enter digits. A dial plan example used as input to the parser and the output dial plan generated are shown in Figure 8.

The other task carried out by the proposed platform is to notify the administrator about the use of deprecated or potentially unsafe functions. This is done while reducing the rule which has the nonterminal *functioncall* on the left side of the production rule. At this moment, the lexeme of the token NAME is obtained, and the parser checks if it is included in a configurable bag of words. This contains the names of all the functions considered unsafe or deprecated. If the lexeme is included in the bag of words, a warning message is shown to the administrator, but the dial plan is not modified. If the lexeme is not included in the bag of words, the function being processed is not considered unsafe or deprecated and the parser will copy it to the output dial plan.

5.3. Hardening Assessments. In order to minimize risks that may compromise the security of Asterisk, the proposed platform assesses the configuration of the PBX and compares it with a predefined set of rules, recommendations, and good practices that help reduce the exposition and possibility of

```
⟨dialplan⟩       ::=  ⟨contexts⟩
⟨contexts⟩       ::=  ⟨contexts⟩ ⟨context⟩ | ⟨context⟩
⟨context⟩        ::=  NAME ⟨prioritiesOpt⟩
⟨prioritiesOpt⟩  ::=  ⟨prioritiesOpt⟩ ⟨priority⟩ | λ
⟨priority⟩       ::=  EXTEN ARROW ⟨extension⟩ , CTEINT , ⟨functioncall⟩
                      | SAME ARROW n , ⟨functioncall⟩
⟨functioncall⟩   ::=  NAME ( ⟨argsOpt⟩ )
⟨extension⟩      ::=  CTEINT | ⟨pattern⟩
⟨pattern⟩        ::=  _ ⟨parts⟩ ⟨wildOpt⟩
⟨parts⟩          ::=  ⟨parts⟩ ⟨part⟩ | λ
⟨part⟩           ::=  CTEINT | [ ⟨constants⟩ ] | X | Z | N
⟨constants⟩      ::=  ⟨constants⟩ - CTEINT | CTEINT
⟨wildOpt⟩        ::=  ! | . | λ
```

Figure 7: Dial plan grammar used by the parser.



```
[from-internal]
exten => _NN45[56-78]XX!,1,Dial(PJSIP/${EXTEN},10)
same => n,Wait(1)
same => n,Playback(gadget)
same => n,Hangup()
```

```
[from-internal]
exten => _NN45[56-78]XX!,1,Set(SAFE_EXTEN=${FILTER(0-9,${EXTEN})})
same => n,Dial(PJSIP/${SAFE_EXTEN},10)
same => n,Wait(1)
same => n,Playback(gadget)
same => n,Hangup()
```

Figure 8: Modifications made by the parser to avoid dial plan injection attacks.

suffering an attack. As previously discussed, the configuration files of Asterisk drive the operation of the PBX. Each of them is composed of different sections and parameters that can be modified, leading to different behaviours and exposures. For this reason, each configuration file has its own recommendations and alerts. The analysis carried out by the platform is guided by the administrator using the web control panel. It encompasses three aspects. Firstly, the general configuration of the Asterisk PBX is analyzed. Secondly, a similar process is performed to assess the configuration of the AMI interface used to connect to Asterisk. Finally, the configuration of each signalling channel is also analyzed. In this work, the platform is focused only on the SIP and PJSIP channels, since SIP is the most widespread signalling protocol, although the set of target protocols could easily be extended. As a result of this process, the administrator receives a series of notifications alerting about the parameters that should be modified to improve the hardening of Asterisk.

The verification of the general configuration of the system involves the analysis of the *asterisk.conf* file, since this file allows for adjusting several parameters that affect the operation of Asterisk as a whole. To do so, several parameters are checked. Firstly, the maximum number of allowed concurrent calls supported by the PBX (`maxcalls`) is checked. If this number exceeds the default value (10), the platform raises an alert. An inappropriate configuration of this parameter may have harmful consequences when combined with a dial plan injection attack, since it serves as the last barrier to control the number of simultaneous calls through the PSTN. Secondly, the platform notifies whether the parameter `live_dangerously` is activated, since it enables the execution of dangerous dial plan functions, such as `SHELL`, from external interfaces which may lead to privilege escalation. Finally, the platform raises an alert if any of the parameters included in the `files` section differ from default values. This section includes options related to the remote control of Asterisk, so configuring them incorrectly may have severe consequences on the security of the PBX.

The second aspect to be assessed is the configuration of the AMI interface, which is stored in the *manager.conf* file. This file contains a list of users that are allowed to use the AMI, their passwords and privileges, IP addresses where they can connect to, and so on. First, the platform raises an alert whenever the AMI is accessible without using TLS. In addition, regardless of whether TLS is configured or not, the platform notifies two more circumstances as dangerous: the use of well-known default ports, 5039 for TLS and 5038 for non-TLS communications, and the binding of the AMI interface with all the network interfaces of the Asterisk PBX, instead of just one particular interface. Furthermore, alerts are also raised if the Asterisk PBX is configured to allow multiple simultaneous connections from the same account, the timeout before a session is discarded exceeds the default value, or there are no restrictions in the IP addresses defined for any user account of the AMI. Some of these checks are also performed on the configuration of the Secure Shell (SSH) management interface in the *sshd_config* file. Specifically, the platform raises an alert in any of the following four cases: default port used, no restrictions in the listening address, login as root user enabled, and login without using password allowed.

Finally, regarding the signalling protocol, both for the SIP (*sip.conf*) and the PJSIP (*pjsip.conf*) channels, the platform issues a warning if the selected transport protocol does not encrypt communications. Similarly, the administrator receives an alert if the port used is the default port (5060) and if the Asterisk PBX is listening for SIP traffic on all its network interfaces. These two cases are also reported when SIP traffic is carried over TLS. The platform also analyzes the SIP channel to check whether an account scan would be able to infer if an username exists or not and whether unregistered callers are allowed to call through the Asterisk PBX. Furthermore, the configuration of the access control lists in the *acl.conf* file is also assessed. The platform raises an alert under two circumstances: if there are no ACLs defined and if any defined ACL does not include at least one deny rule, since the default behaviour in Asterisk dictates that the SIP traffic, even though it has no match with any rule, is permitted.

5.4. Platform Management. The functionality of the diagnosis and hardening platform is easily extensible and highly customizable. To do so, the database serves as a driving element of the system, since the behaviour of the controller can be instructed according to the entries of several control tables. Therefore, the controller only performs the actions that correspond to directives defined in the database. The administrator can make adjustments to operational instructions in the database or create new ones through the web control panel hosted in the controller. This management procedure prevents premature obsolescence of the platform, since new configurations and conditions can be easily included during diagnosis and hardening assessments.

The administrator can manage the following aspects of the platform. First, the monitoring and processing of new security events can be enabled, increasing the set shown in Table 1. The risk categorisation of events can be modified. Similarly, the number of occurrences and the time intervals to trigger reactive actions can also be configured. Furthermore, events raised in newer Asterisk versions can be incorporated to the platform. Second, future deprecated or compromised methods and applications that should not be used in the dial plan can be added to the bag of words described in Section 5.2. Therefore, extending the catalogue of threats is not only simple but also flexible. Third, additional bad practices to avoid, as well as hardening considerations related to Asterisk configuration files, can also be included.

Finally, these changes and extensions in the functionality of the platform can be done in real time. This will not disrupt the execution of any of the entities of the platform since the database is queried by the controller before running both the diagnosis and the hardening assessments ordered by the administrator.

6. Experimentation

Ideally, an Asterisk PBX is deployed in a real corporate site using a private IP address behind a NAT and a firewall or a session border controller. Thus, Asterisk manages outgoing and incoming calls between the LAN and other networks across the Internet. In this scenario, Asterisk is exposed to the set of external threats explained in Section 4.1. Therefore, the performance of the proposed platform when mitigating them can be evaluated. In order to perform the experimentation, the simplified VoIP testbed environment shown in Figure 9 is used.

Legitimate VoIP users (green striped circles) are emulated using several Grandstream GXP2170 VoIP phones with valid SIP credentials. External attackers are emulated using bots (solid red circles). All platform components and bots are installed in commodity computers with identical characteristics. Table 2 summarizes the characteristics of the computers used. Finally, a Cisco 2960 Gigabit Ethernet switch is used to emulate the network that interconnects the proposed platform, legitimate users, and bots. Platform components (grey squares) are installed on dedicated devices. Asterisk is deployed on a single computer, and the other components of the platform (controller, receiver, and database) are co-located in another computer.

Bots are customized SIP traffic injectors built on top of SIP Vicious, a set of penetration testing tools for auditing SIP-based VoIP systems [24]. These bots continuously perform account scans on Asterisk from multiple dynamically generated source IP addresses within the network, while ensuring that these IP addresses are different from those used by platform components and legitimate users. The network configuration uses a/16 subnet mask in order to have a large number of IP addresses available.

In the experimentation, the number of bots trying to login to Asterisk is varied sequentially. The platform is configured to classify as attackers users who send a certain number of consecutive unauthorized SIP INVITEs (n_{ua}), that is, they are trying to log in the PBX with incorrect credentials, during a one-second time interval. Three aspects are assessed on the platform: the ability to block attackers when performing account scans against the PBX; an analysis of the performance of the platform under these attack circumstances; and the impact of using the platform on the QoS of ongoing calls between legitimate users. Table 3 shows the parameters used during most of the tests.

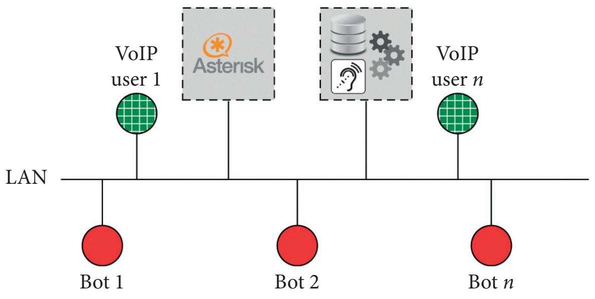


Figure 9: Simplified testbed environment.

Table 2: Characteristics of bots and platform components.

Processor	Intel Core i3 4150 3.5 GHz
Memory	8 GB SDRAM DDR3
NIC	1 Gb Ethernet
Hard disk	500 GB SATA
Operating system	Kali Linux (bots) Windows 10 (platform)
PBX	Asterisk 15, Ubuntu 18.04 LTS

Table 3: Test settings.

Test duration (s)	600	
Number of repetitions	3	
Scan account duration (messages)	(μ_m, σ_m)	(100, 20)
Interval between messages (s)	(μ_i, σ_i)	(0.1, 0.05)
Interval between scans (s)	(μ_s, σ_s)	(0.1, 0.05)
LAN available bandwidth (Mbps)	100	

7. Results

Figure 10(a) shows the number of attackers blocked in real time by the platform according to the number of bots running simultaneously. Several scenarios were tested varying the number of allowed consecutive unauthorized login attempts (n_{ua}) during a one-second interval. As the platform detects malicious account scan attacks, the number of source IP addresses blocked by the platform grows linearly. The blocking ability of the platform does not depend on the value of n_{ua}. Figure 10(b) shows the relation between the number of SIP INVITE requests received from attackers and the number of unauthorized responses sent back. The area between these two curves can be considered as the SIP resources saved in Asterisk, since it corresponds to the number of SIP responses that the platform avoids sending back to attackers. As expected, the lower n_{ua}, the higher the resource saving is achieved.

In Figure 11(a), the global CPU usage by Asterisk is plotted as the number of bots grows. Two different deployments are compared, the proposed platform with several values of n_{ua} and a bare Asterisk. As observed in the figure, there is a linear relation between the CPU usage and the number of bots. The CPU usage is higher when the platform is not deployed, since every single SIP request is processed and answered. The different values that n_{ua} takes barely affect the CPU usage, although the trend shows that this could vary more noticeably for a high number of concurrent bots.

Moreover, the memory consumption by Asterisk according to both deployments is shown in Figure 11(b). Memory consumption is hardly affected by the growing number of bots. The proposed platform uses more memory due to the interactions between Asterisk and other components, such as the receiver and the controller. However, this increment can be considered as negligible. In addition, there is not significative difference when varying the number of allowed unauthorized login attempts.

Ideally, the total number of unauthorized responses sent by Asterisk could be calculated by multiplying the number of blocked attackers by n_{ua}, since an attacker is detected and blocked after n_{ua} consecutive unauthorized login attempts. However, in practice, more responses are sent for two reasons. First, Asterisk does not enqueue every SIP INVITE received while waiting for the controller to decide whether or not it should be answered, so the establishment time of calls between legitimate users is not affected. Second, there is a communication and processing delay where extra SIP INVITEs may be processed between the detection of the attacker and the reconfiguration of Asterisk.

Figure 12(a) shows the difference between the ideal number of responses that should be sent by Asterisk to the blocked attackers in each test case and the total number of responses that have actually been sent. The ideal count of unauthorized responses that should be sent is calculated by multiplying the number of source IP addresses blocked by n_{ua}. As can be seen, the difference grows more sharply when the volume of SIP requests and the number of attackers is large, regardless of the value of n_{ua}. Therefore, the overhead in the PBX grows with the number of concurrent attackers, so the number of extra SIP INVITEs that are processed by Asterisk before blocking an attacker increases.

Therefore, the platform effectiveness ratio can be calculated by dividing the ideal count of unauthorized responses that should be sent by the real count of unauthorized responses actually sent. Figure 12(b) shows the platform effectiveness according to different n_{ua} values. Horizontal lines represent, for each different value of n_{ua}, an effectiveness value that would correspond to one extra SIP INVITE in average that is processed by Asterisk for every attacker ($n_{ua}/(n_{ua}+1)$). Figure 12(b) shows that the platform effectiveness remains above sixty percent in the worst test scenarios using commodity hardware. The exception to this is $n_{ua} = 1$, where extra SIP INVITEs processed imply a large penalty in the effectiveness ratio.

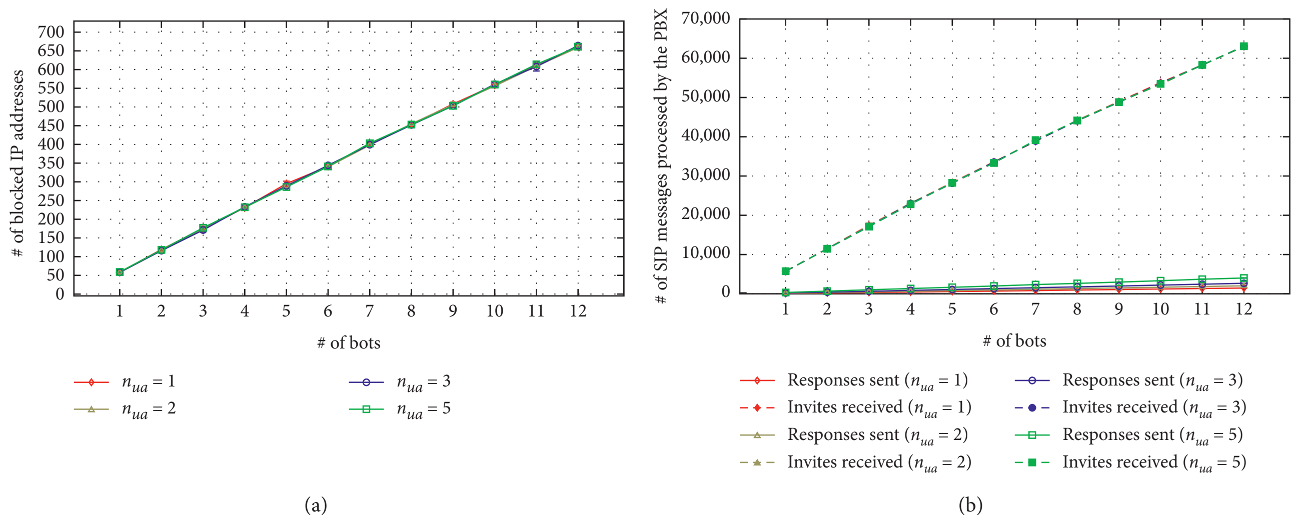


FIGURE 10: Account scan attack detection. (a) Source IP addresses blocked in real time. (b) SIP resources saved in the Asterisk PBX.

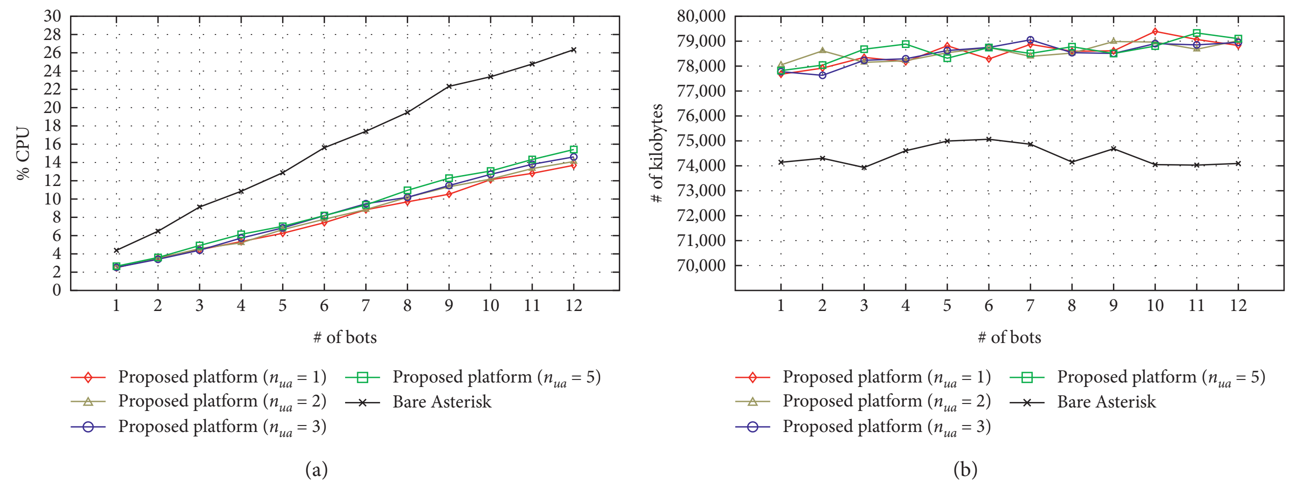


FIGURE 11: Resource consumption comparison between bare Asterisk and proposed platform. (a) CPU consumption. (b) Memory consumption.

Several tests to assess the impact of the proposed platform on QoS have been carried out. The test on the SIP registration process does not have a specific duration but consists of five sequential attempts by different legitimate users to successfully register in Asterisk using valid credentials. As in previous scenarios, both the proposed platform and a bare Asterisk are being attacked by a varying number of concurrent bots. Figure 13 shows a comparison of the time needed to complete a SIP registration process on Asterisk for a legitimate VoIP user. As the number of bots increases, the proposed platform requires a slightly reduced registration time than a bare deployment. This fact confirms that the proposed platform reduces the use of SIP resources in Asterisk when malicious account scans are performed concurrently. This is more noticeable for lower values of n_{ua}.

Finally, the proposed platform is compared to a bare Asterisk deployment when performing a VoIP call between legitimate users. The VoIP calls lasted one minute and were repeated three times with $n_{\text{ua}} = 3$. Figure 14(a) shows the latency in the calls between two VoIP users as the number of bots increases. Similarly, Figure 14(b) plots the jitter measured in these calls. As can be seen, both deployments have no impact on the quality of the calls, causing a latency of 20 ms and an average jitter of 12 ms, without significant differences between the proposed platform and a bare Asterisk deployment.

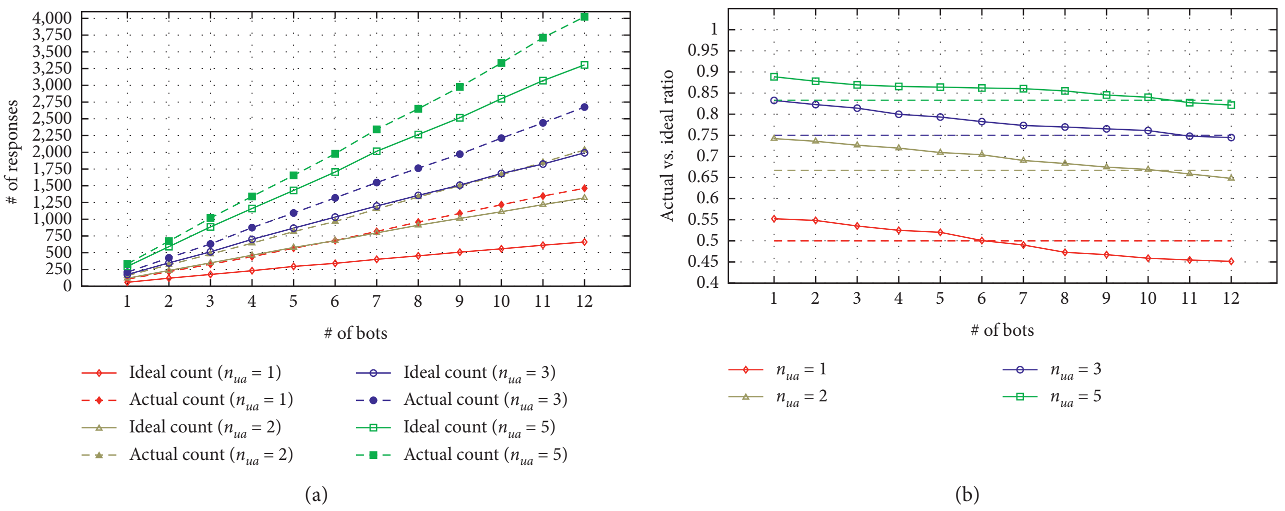


FIGURE 12: Effectiveness of the proposed platform. (a) Unauthorized responses sent. (b) Effectiveness ratio.

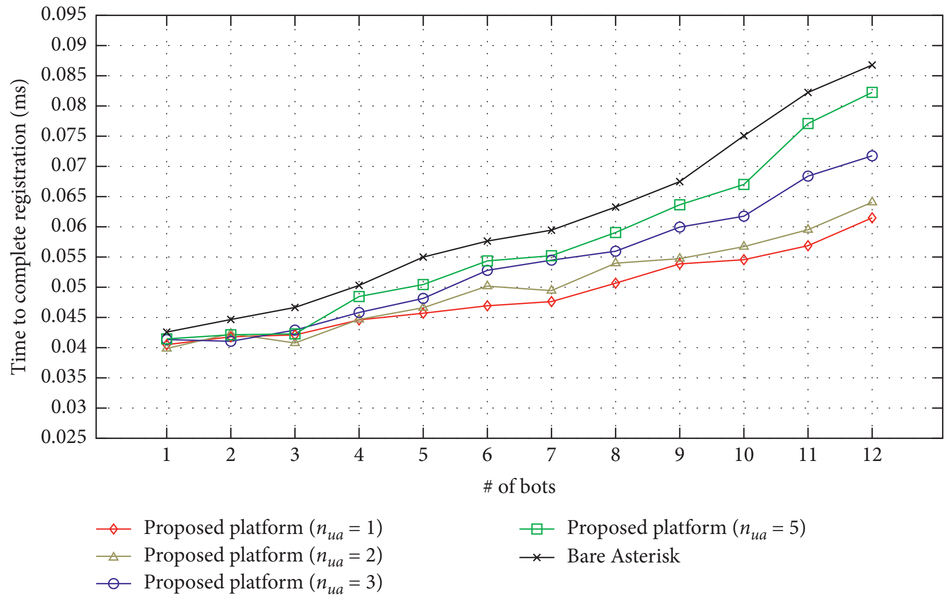


FIGURE 13: Legitimate user registration time.

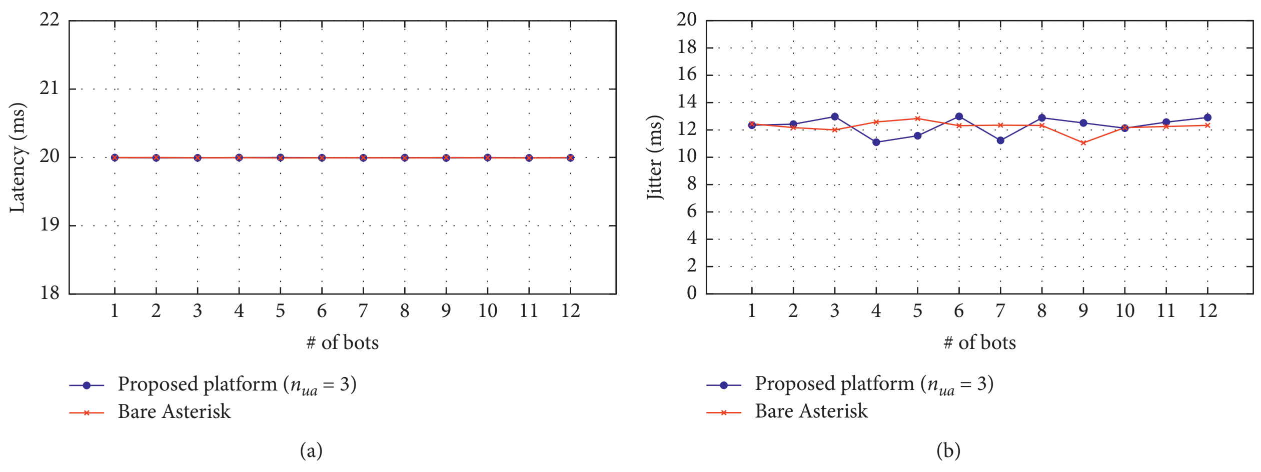


FIGURE 14: Comparison of VoIP call: bare Asterisk and proposed platform. (a) Latency in VoIP calls. (b) Jitter in VoIP calls.

8. Conclusions

In this paper, a diagnosis platform for discovering vulnerabilities and security breaches in the configuration of an Asterisk PBX is presented. The platform takes reactive actions in order to reconfigure the Asterisk PBX in real time, reducing the impact of external threats. Thus, account scan attacks are mitigated by updating access control rules in Asterisk without service disruption. In addition, the platform is able to parse and fix the dial plan to guarantee that neither dial plan injection attacks can be conducted nor functions considered deprecated or potentially unsafe be used. The proposed platform can be easily managed and extended.

The platform is composed of four components: an Asterisk PBX configured to send operational events through the AMI interface; a receiver that collects the security events raised from Asterisk; a relational database for further processing of these events; and a controller that reacts after every incoming security event stored in the database. The controller is able to autonomously update the configuration of the Asterisk PBX through the AMI interface. It can also be manually instructed by the administrator to perform several on-demand diagnosis and hardening assessments such as dial plan parsing and fixing.

Tests show that the proposed platform fulfils its purpose. It reduces the exposure of Asterisk when facing account scan attacks, leading to a more efficient use of SIP resources, without penalising the quality of service of ongoing calls between legitimate VoIP users.

The main drawback of the platform is its effectiveness under account scan attacks. The communication and processing delay of the security events, raised from Asterisk and handled by the controller, leads to a slight increment in the number of unauthorized responses sent when compared to the ideal. This minor issue can be alleviated by co-locating both the controller and the receiver with the Asterisk PBX or fully integrating them as Asterisk modules.

Future work will be focused on using machine learning techniques in order to detect potentially dangerous calls made by legitimate users by identifying a suspicious behaviour in real time.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was partially funded by the project RTI2018-094849-B-I00 of the Spanish National Plan for Research, Development and Innovation.

References

[1] J. Packer and W. Reuschel, “VoIP accessibility: a usability study of voice over internet protocol (VoIP) systems and a survey of VoIP users with vision loss,” *Journal of Visual Impairment & Blindness*, vol. 112, no. 1, pp. 47–60, 2018.
[2] S. Karapantazis and F.-N. Pavlidou, “VoIP: a comprehensive survey on a promising technology,” *Computer Networks*, vol. 53, no. 12, pp. 2050–2090, 2009.
[3] A. Martin, E. Gamess, D. Urribarri, and J. Gómez, “A proposal for A high availability architecture for VoIP telephone systems based on open source software,” *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 9, pp. 1–11, 2018.
[4] A. Montazerolghaem, S. A. Hosseini Seno, M. H. Yaghmaee, and F. Tashtarian, “Overload mitigation mechanism for VoIP networks: a transport layer approach based on resource management,” *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 6, pp. 857–873, 2016.
[5] K. Suwannaraj and S. Boonkrong, “Efficiency improvement of outbound call route selection on IP-PBX using KNN with USSD,” *Research Journal of Applied Sciences*, vol. 9, no. 9, pp. 556–564, 2014.

[6] S. Senthil Kumar, B. Dhivyalekshmi, S. Preethi, and P. Rengaraju, “PBX implementation in LAN using Asterisk open source software,” *International Journal of Applied Engineering Research*, vol. 10, no. 55, pp. 66–69, 2015.

[7] C. K. Gohel and K. I. Lakhtaria, “Implement VoIP based IP telephony with open source asterisk architecture,” *International Journal of Interdisciplinary Telecommunications and Networking*, vol. 2, no. 1, pp. 1–11, 2010.

[8] O. Dabbebi, R. Badonnel, and O. Festor, “An online risk management strategy for VoIP enterprise infrastructures,” *Journal of Network and Systems Management*, vol. 23, no. 1, pp. 137–162, 2013.

[9] A. D. Keromytis, “A comprehensive survey of voice over IP security research,” *IEEE Communications Surveys & Tutorials*, vol. 14, no. 2, pp. 514–537, 2012.

[10] R. Ackermann, M. Schumacher, U. Roedig, and R. Steinmetz, “‘Vulnerabilities and security limitations of current IP telephony systems,’” in *Proceedings of the 6th Conference on Communications and Multimedia Security*, pp. 53–66, (CMS’01), Darmstadt, Germany, May 2001.

[11] D. Geneiatakis, T. Dagiuklas, G. Kambourakis et al., “Survey of security vulnerabilities in session initiation protocol,” *Institute of Electrical and Electronics Engineers Communications Surveys & Tutorials*, vol. 8, no. 3, pp. 68–81, 2006.

[12] S. Ehlert, D. Geneiatakis, and T. Magedanz, “Survey of network security systems to counter SIP-based denial-of-service attacks,” *Computers & Security*, vol. 29, no. 2, pp. 225–243, 2010.

[13] V. Ganesan and M. Msk, “A scalable detection and prevention scheme for voice over internet protocol (VoIP) signaling attacks using handler with Bloom filter,” *International Journal of Network Management*, vol. 28, no. 2, pp. 1–18, 2018.

[14] A. M. Hagalisletto and L. Strand, “Designing attacks on SIP call set-up,” *International Journal of Applied Cryptography*, vol. 2, no. 1, pp. 13–22, 2010.

[15] U. U. Rehman and A. G. Abbasi, “Secure layered architecture for Session Initiation Protocol based on SIPSSO: formally proved by Scyther,” in *Proceedings of the 12th International Conference on Information Technology - New Generations (ITNG)*, pp. 185–190, Las Vegas, NV, USA, April 2015.

[16] S. Jung, “CAPTCHA-based DDoS defense system of call centers against zombie smart-phone,” *International Journalof Security and Its Applications*, vol. 6, no. 3, pp. 29–36, 2012.

[17] M. Voznak and F. Rezac, “Threats to Voice over IP communications systems,” *WSEAS Transactions on Computers*, vol. 9, no. 11, pp. 1348–1358, 2010.

[18] R. K. Lomotey and R. Deters, “Intrusion prevention in Asterisk-based telephony system,” in *Proceedings of the IEEE International Conference on Mobile Services (MS 2014)*, pp. 116–123, Anchorage, AK, USA, July 2014.

[19] B. Cha, J. Kim, H. Moon, and S. Pan, “Global experimental verification of Docker-based secured mVoIP to protect against eavesdropping and DoS attacks,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2017, no. 1, pp. 1–14, 2017.

[20] V. Ganesan, N. Shalini, and M. Manikan, “Navie bayes intrusion classification system for VoIP network using honeypot,” *International Journal of Engineering*, vol. 28, no. 1, pp. 46–53, 2015.

[21] B. Materna, “Proactive security for VoIP networks,” *Information Systems Security*, vol. 15, no. 2, pp. 16–21, 2006.

[22] M. Jouini, L. B. A. Rabai, and A. B. Aissa, “Classification of security threats in information systems,” *Procedia Computer Science*, vol. 32, pp. 489–496, 2014.

[23] “Asterisk list of events,” https://wiki.asterisk.org/wiki/display/AST/Asterisk+15+AMI+Events, 2017.

[24] *SIPVicious Tool*, https://github.com/EnableSecurity/sipvicious, 2015.