SIP Proxies: New Reflectors in the Internet

Ge Zhang, Jordi Jaen Pallares, Yacine Rebahi, Simone Fischer-Hübner

SIP Proxies: New Reflectors in the Internet

Ge Zhang[1], Jordi Jaen Pallares[2], Yacine Rebahi[2],
Simone Fischer-Hübner[1]

[1] Karlstad University, Karlstad, Sweden
[2] Fraunhofer FOKUS, Berlin, Germany

Abstract. To mitigate identity theft in SIP networks, an inter-domain authentication mechanism based on certificates is proposed in RFC 4474 [10]. Unfortunately, the design of the certificate distribution in this mechanism yields some vulnerabilities. In this paper, we investigate an attack which exploits SIP infrastructures as reflectors to bring down a web server. Our experiments demonstrate that the attacks can be easily mounted. Finally, we discuss some potential methods to prevent this vulnerability.

1 Introduction

The current most widely used authentication method for SIP is based on HTTP hash digest mechanism. Therefore SIP users need to share secrets (e.g., user account, password) with their home domains for message source authentication. However, such shared secrets are usually not distributed to other domains. Thus, identity misuse in multi-domain environments is possible, and accordingly will lead to some security problems such as VoIP SPAM and caller impersonation. To address this problem, J. Peterson, et al., [10] proposed a certificate based inter-domain authentication mechanism: An originator SIP domain signs an outgoing message with its private key and a recipient SIP domain verifies the signature of the incoming message. Thus, the recipient SIP domain needs to have the certificate of the originator domain for verifying messages. Nevertheless, there is a serious problem in the designed certificate distribution scheme: A recipient SIP proxy should download the certificate according to a URL address specified in the SIP message. *However, at the moment of certificate downloading, the SIP message is still unverified and untrusted.* Therefore, this mechanism offers an opportunity for an attacker to let a SIP proxy connect to an arbitrary remote host. In our previous work [15], we demonstrated an attack which exploits the time for certificate downloading to block a victim SIP proxy. In this paper, we aim to investigate another attack: The attacker can collect a list of SIP proxies as reflectors, then use the reflectors to flood a victim web server. The results of our experiments demonstrate that this attack can cause a serious impact, especially on a web server which supports HTTPS protocol.

The idea behind this attack is that a SIP proxy needs to "pull" a certificate according to a unverified request. To prevent the problem, we modified distribution method to "push" a certificate: An originator should actively provide its

domain certificate. Our contribution in this paper is two-fold: First, we investigate a reflector DoS attack generated by the certificate distribution scheme described in RFC 4474. Secondly, we address improved schemes to deal with the threat.

The current paper is organized as follows: Section 2 provides an overview of SIP as well as its inter-domain authentication mechanism. Section 3 investigates the threats caused by the inter-domain authentication mechanism. Countermeasure solutions are discussed in Section 4. We summarize related work in Section 5, and in Section 6, we provide a conclusion.

2 Background of SIP and RFC 4474

SIP [12] is a text-based protocol designed to establish, modify, or terminate a session (e.g., VoIP calls) between two or more partners. Several SIP networking components are essential to be introduced: User Agent (UA), proxy server, registrar server. A UA represents a communication endpoint. According to its role in a communication, a UA can be a UA client (UAC) which initializes a SIP message, or a UA server (UAS) which receives a SIP request and makes a response. A proxy server (or proxy) forwards SIP messages between different SIP components in the network. A registrar server is where users login and announce their availability in the network. Generally, some security checks (e.g., authentication) are enforced on a proxy. SIP users are identified by their Uniform Resource Identifiers (URI) [2]. A URI consists of a domain name and a user name (e.g., `sip:Alice@kau.se`). The format of a SIP message is similar to HTTP protocol, with message headers and a message body with its corresponding values (e.g, *From: Alice@kau.se* is to denote the sender of a message). SIP messages are distinguished as two categories: SIP requests (e.g., REGISTER, INVITE, ACK, BYE, etc) and SIP responses (1xx, 2xx, etc). In this paper, we focus on inter-domain requests. We define that a SIP request transverses over different SIP domains as *inter-domain request*. The proxy which initializes an inter-domain request is defined as *originator proxy*, and its domain is named as *originator domain*. While the destination proxy is named as *recipient proxy*, within a domain defined as *recipient domain*.

VoIP attackers frequently use fraud SIP identities in order to be untraceable. As a countermeasure, RFC 3261 [12] proposed a HTTP digest based authentication mechanism using shared secrets (e.g., user name and password). Unfortunately, this solution does not work in multi-domain environments since the secretes are usually not distributed. In this way, attackers can easily fraud their SIP identities in multi-domain environments without being detected. To prevent such a problem, an inter-domain authentication mechanism based on certificates has been proposed in RFC 4474 [10] and illustrated in Figure 1. Let us say that a caller sends a calling request to a callee in a multi-domain scenario. The originator domain signs the digest of this request with its private key. The generated signature is encoded within a new header filed *Identity* appended to the original request. Moreover, the originator domain attaches another new header field

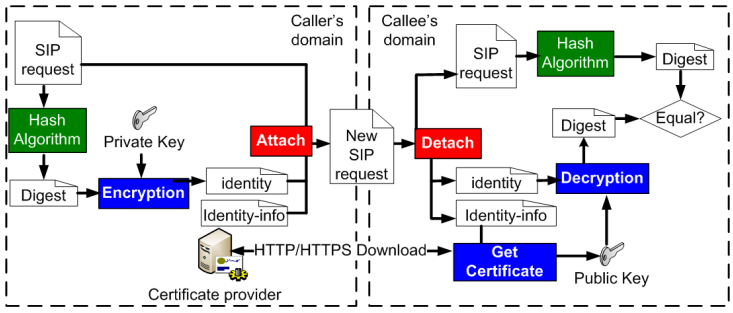


Fig. 1. The inter-domain authentication mechanism defined in RFC 4474

Identity-info, which contains the Uniform Resource Locator (URL) [3] indicating where the certificate can be downloaded from[3] as well as authentication parameters (e.g., the algorithm used for verification). This modified request is then forwarded to the recipient domain. Thus, the recipient domain will first check whether the certificate for the originator domain can be found locally (e.g., it has already been cached). If the certificate is not cached, the proxy has to download it according to the URL given in the *Identity-info* field. If the downloading does not succeed or the downloaded certificate is invalid, the recipient domain will neglect this request and reply with a negative response. Otherwise, the proxy uses the public key extracted from the certificate to verify the signature in the *Identity* header field. The request will be finally forwarded to the callee only if the verification is successful.

3 Reflecting attacks using SIP proxies

We assume that all the SIP proxies mentioned along this paper are implemented according to the authentication scheme specified in RFC 4474. A serious problem of this authentication scheme is the fact that the recipient proxy cannot know whether an incoming inter-domain request is spoofed or not until the authentication operation takes place. To do so, the recipient proxy needs the corresponding certificate. If the certificate is not cached, the proxy has to download it from the location specified in this *Unverified* request. In this way, an attacker can cheat one or many proxies to let them connect with any host for "certificate downloading". This vulnerability can lead to DoS attacks on a web server, despite of whether this web server really supports domain certificate downloading or not. In this paper, we focus on a system model which is a large-scaled network consists of numerous SIP proxies and web servers (e.g., the Internet). We assume that the attacker can connect to the network and craft SIP requests to a number of SIP proxies. The goal of the attacker is to reduce the availability of a targeted web server.

[3] According to RFC 4474, the URL indicated in the *Identity-info* header field and the originator domain indicated in the *From* header field must be matched.

Attack description DoS attacks generally take advantage of unbalanced resource occupation between client side and server side. To understand the situation of resource consumption in the given context, we performed a measurement with a testbed configured in Figure 2.

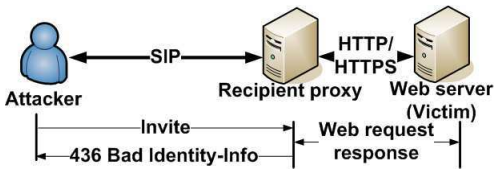


Fig. 2. The testbed for measuring resource consumption

The testbed consists of three networking components as follows:

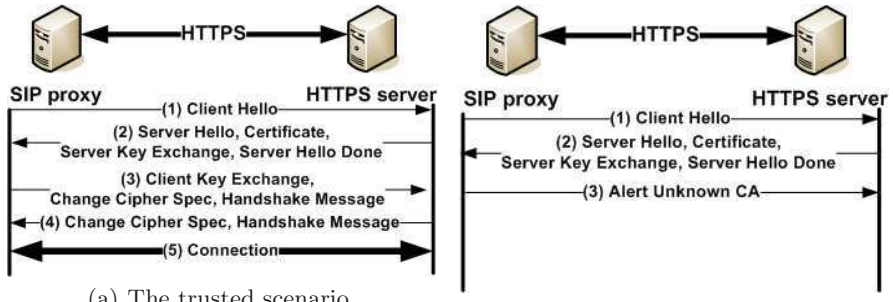
1. A web server: In this test, we employ Apache [1], a widely deployed web server program. We configure it to support both HTTP and HTTPS connections. To enable HTTPS connections, we generate a self-signed X.509 RSA certificate on the web server[4]. The size of this HTTPS certificate is 806 bytes with 1024 bits public key.
2. A SIP proxy: We employ SIP Express Router (SER) [13] as our SIP proxy, which is configured with 16 parallel process queue.
3. An attacker: The attacker generates crafted inter-domain SIP requests. The attacker can designate the scheme name (http://, or https://) of the *Identity-info* URL in a SIP request. The attacker is implemented using SIPp [14], an open source SIP traffic generation tool.

Hardware: Each component mentioned in this paper is independently established on a Pentium 4 machine with 512 Mb RAM running the linux Ubuntu operating system, with 100 Mbps local network access.

Our measurement investigates the CPU overhead on the web server and the attacker respectively for attacking. The purpose of this test is to find out the difficulty of mounting this attack. A DoS attack targets resources of the victim. Moreover, the DoS attack is reasonable for the attacker only if it is not required to spend a lot of resources in equipment to achieve the task. During the following test, we investigate the attacker and the proxy CPU utilization according to different attacking rate configured for the attacker. Our measurement is repeated for three different circumstances:

1. HTTP scenario: The attacker floods the proxy with requests in which the *Identity-info* header fields begin with the input "http://". In this way, the proxy will need to connect to the web server using HTTP protocol.

[4] This certificate is only used for web server authentication during HTTPS handshake, not for SIP inter-domain authentication purpose. To distinguish it from the certificate for SIP authentication, we call it as HTTPS certificate in the rest of this paper



(a) The trusted scenario

(b) The untrusted scenario

Fig. 3. Establishing a HTTPS connection between the proxy and the web server

2. HTTPS (Trusted) scenario: The attacker sends requests to the proxy in which the *Identity-info* header fields begin with scheme name "https://". The proxy first receives the certificate from the web server and it trusts this self-signed HTTPS certificate (e.g., it is from a trusted root CA). Therefore, the proxy will exchange session keys and build HTTPS connections with the web server. The procedure is illustrated in Figure 3(a).
3. HTTPS (Untrusted) scenario: Similarly, the attacker send the proxy with requests in which the *Identity-info* header fields begin with scheme name "https://". However, the proxy does not trust the self-signed HTTPS certificate of the web server in this scenario. In this way, the handshake is interrupted after the SIP proxy checks the HTTPS certificate sent from the web server. Thus the HTTPS connection and downloading cannot be successful. We depict this procedure in Figure 3(b).

Figure 4 compares the average CPU utilization on the attacker and the web server with different attacking rates and scenarios. We can hardly find any variability of the CPU utilization on the attacker whatever the used attacking rate and the scenario type. On the other hand, The CPU utilization on the web server stayed 4% or so for the HTTP scenario. The value increases from around 4% to almost 74% for the HTTPS (trusted) scenario. However, this value surges from 4% to around 72% for the HTTPS (Untrusted) scenario. With an increasing attack rate, the CPU utilization on the web server surges for the HTTPS scenarios. The CPU utilization on the web server is much higher than it in the attacker in the HTTPS scenario. This result is mainly due to the random number generating and the cryptographical operations, which consume a lot of CPU resources. As shown in the results, it is easy to consume a significant resources of a HTTPS web server by using little resources on the attacker side. This unbalanced situation might be due to the following reasons:

1. HTTP scenario: TCP was specifically designed to provide a reliable end-to-end communication with a "flow control" method. However, in return, the "flow control" method consumes additional bandwidth. On the other

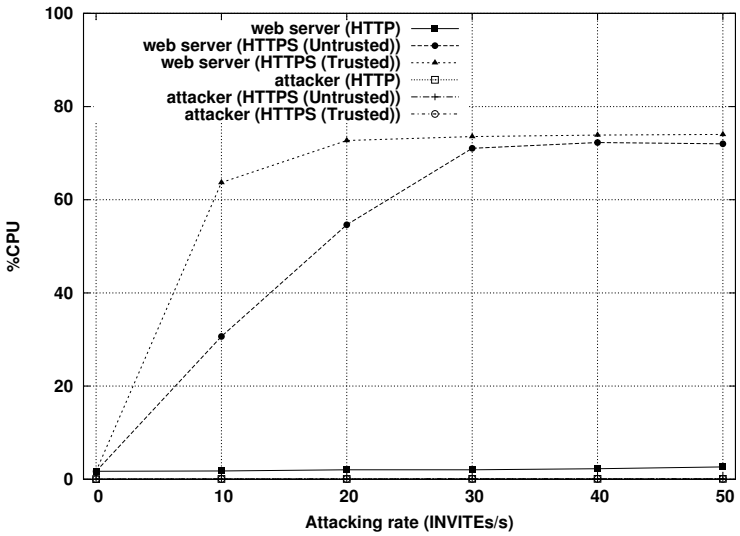


Fig. 4. The consumption of CPU resources on different components

hand, UDP, without a "flow control" method, is more efficient than TCP. SIP communications can be built on either TCP or UDP. During the test, the attacker sends requests to the SIP proxy over UDP while the SIP proxy builds HTTP connections with the web server over TCP. That is why that a little more resource consumed on the web server than on the attacker. Anyway, the gap is rather small.

2. HTTPS (Trusted) scenario: According to Figure 3(a), it is clear that much more resources are needed for building up a HTTPS connection since it spans over several steps. For example, to exchange session keys with the proxy, the web server has to perform a lot of computation that occupies a big part of the CPU resources.
3. HTTPS (Untrusted) scenario: Although the connection cannot be setup in this scenario, the CPU utilization on the web server is still high. From Figure 3(b), the server key exchange is performed in step 2. The proxy cancels the transaction in step 3 with an alert message, however, even in this case, the server has to perform a key exchange operation. By default, it is an RSA key exchange that takes place by creating a temporary RSA key pair. This procedure consumes a lot of CPU resources.

In this way, we assume that an attacker has a target HTTPS web server with URL: `https://www.victim.com`. To bring it down, an attacker just needs to select a list of SIP proxies in the network as reflectors. Then, the attacker floods the proxies with numerous crafted SIP requests in which the *identity-info* header fields include the same value: `https://www.victim.com/[random_string]`. As a consequence, the SIP proxies will continuously build connections with the victim web server and the CPU resources of the web server will be depleted.

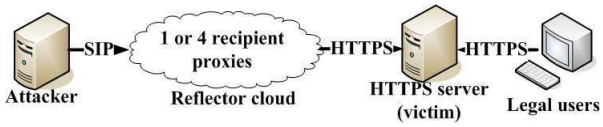
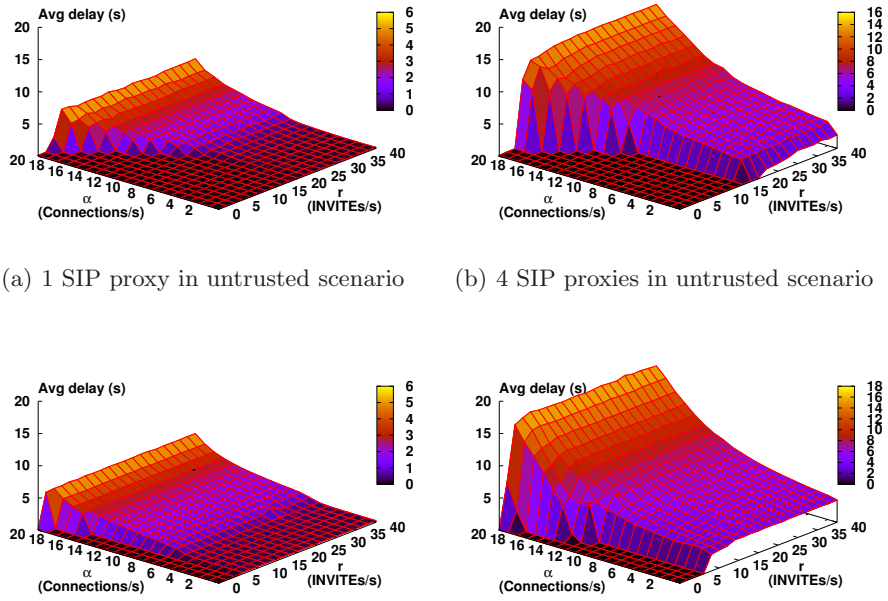


Fig. 5. The testbed for distributed reflector attacks on HTTP web servers



(a) 1 SIP proxy in untrusted scenario

(b) 4 SIP proxies in untrusted scenario

(c) 1 SIP proxies in trusted scenario

(d) 4 SIP proxies in trusted scenario

Fig. 6. The victim web server response delays for different parameters

Experiments setup To investigate the attack in detail, we setup our testbed in the way depicted in Figure 5. The testbed consists of the following components:

1. The attacker: The attacker works in the same way as in the previous test. It generates SIP requests with the *identity-info* header field including the URL of the victim's HTTPS web server. The attacking rate r (INVITE/s) can vary from 0 to 40.
2. Reflector cloud: The attacker employs SIP proxies as attacking reflectors. As soon as the SIP proxies receive requests from the attacker, they will try to establish HTTPS connections with the victim web server. In this

experiment, we setup the reflector cloud with either one or four SIP proxies. Please notice that the attacking rate r (INVITEs/s) is for the whole cloud. Therefore, if there is only one proxy in the cloud, then the proxy receives attacking requests at the rate r. However, if there are four proxies in the cloud, then each proxy receives the attacking requests at a rate equal to $r/4$.

3. HTTPS web server: The configuration of the web server is similar to the previous test. However, it only supports HTTPS protocol in this test.
4. Legitimate users: We employ httpref [6], a web server performance measurement tool, to simulate legitimate users, which constantly build HTTPS connections with the victim web server. The performance of the victim web server can be demonstrated by observing the delay of connecting the legitimate users to the web server. The legitimate users can be configured to constantly build HTTPS connections at a rate α (connections/s), varying from 1 to 20.

In the test, we measure the average connecting delay between the legitimate users and the web server. The result is shown in Figure 6. It shows that average delay of the connection between legitimate users and the web server is about few milliseconds without attack (when $r = 0$), no matter what the other parameters are. However, under attack, the delay can be up to 14 (s) or 16 (s) if the attacker selects 4 reflector proxies. By using only one reflectors, the attack impact is not so high but still serious: The maximal delay is around 4-5 (s). Generally, the attacking impact in trusted scenarios is slightly higher than the one in untrusted scenarios. Furthermore, a higher attacking rate and more legal connections will certainly lead to a higher attacking impact on the web server. A connection attempt with 16 seconds delay is usually regarded as unsuccessful for most web browsers. As a result, legitimate users cannot visit a HTTPS web server if the server is under flooding from the reflector proxies. It is also worth to mention that this kind of attack can be easily performed in reality: An attacker first collects a list of SIP proxies in the Internet as reflectors. Then the attacker floods the reflectors with crafted requests containing certificates location at a victim HTTPS server. In this way, the HTTPS server can be brought down due to too many connecting requests from the reflector proxies.

4 Countermeasures

4.1 Unified certificate repository

One solution is to build a global unified certificate repository to store certificates for different SIP domains. Thus, the location of the unified certificate repository can be hard-coded on a SIP proxy, instead of being specified by the requests. In this way, wherever the SIP requests come from, the recipient proxy should download certificates from such a known unified certificate repository. As a result, attackers no longer can arbitrarily specify which host a proxy should contact with. Unfortunately, however, a unified certificate repository is not scalable and might be difficult to be accepted by service providers in the Internet.

4.2 Alternative authentication methods

As alternatives, other security mechanisms at the transport layer or network layer can also be considered as candidates for inter-domain authentication purpose. For example, establishing a Transport Layer Security (TLS)[5] connection or an IPSec[8] tunnel between an originator proxy and a recipient proxy. As for the use of TLS within this context, RFC 4474 mentions that while the creation of a chain of transitive authentication between the originating UA, the local proxy and the remote proxy via TLS may work well in some architectures, transitive trust is inherently weaker than an assertion that can be validated end-to-end. In the case a SIP request is crossing multiple SIP domains, this transitive trust becomes even less effective. RFC 3325[7] proposes a solution to this problem implementing the *P-Asserted-Identity header*. However, this solution only allows hop-by-hop trust between intermediaries, and no end-to-end authentication. In addition to that, it assumes a managed network of nodes with strict mutual trust relationships, an assumption that is incompatible with widespread Internet deployment. Similarly, IPSec cannot assure an end-to-end protection, either.

Furthermore, TLS is connection-oriented and requires also a handshake. In contrast, the mechanism described in RFC 4474 is connectionless and does not require any handshake operation. Moreover, TLS aims at providing confidentiality, integrity and authentication for the entire traffic. Nevertheless, the mechanism proposed by RFC 4474 only focuses on providing authentication and integrity for selected parts of a SIP request. In this way, the RFC 4474 mechanism, a lightweight protection method, introduces less performance overhead than the other methods. With the just discussed benefits related to end-to-end protection and performance, the RFC 4474 mechanism cannot be easily replaced by TLS or IPSec.

4.3 Encoding certificates into the SIP requests

According to RFC 4474, it is the recipient proxy which is responsible for downloading the certificate of the originator proxy. This action is qualified as *pulling* a certificate by the recipient proxy. This *pull* action produces the vulnerability, which makes the victim proxy connect to a networking host arbitrarily set by the attacker. In contrast, we address an alternative way for certificate transferring in which an originator proxy actively *pushes* a certificate to the recipient proxy.

Method Figure 7 illustrates the procedure of a potential "push" scheme. Firstly, we assume that a UA can receive a copy of the domain certificate after registration to this domain. When an originator proxy receives an inter-domain request from its user, the proxy will sign this request, and add the signature into the request as a new header *Identity*. The proxy will also attach the request with a new header *identity-info*. However, different to the one defined in RFC 4474, this *Identity-info* only contains verification parameters (e.g., which algorithm should be used). There is no URL contained in the *Identity-info* headers

any more. The recipient proxy will search the domain certificate of the originator domain in the local cache after getting this request. If the certificate of the originator domain is cached locally, the proxy will then verify the request by using the cached certificate. Otherwise, the recipient proxy will send to the originator proxy a "4XX" response to indicate that a certificate is required for further processing. The UAC in the originating domain will then resend the request along with the domain certificate. Again the originator SIP proxy will sign the request and attach *Identity* and *Identity-info* headers. Finally, the recipient can get the certificate from the request and cache it for a period of time. In this way, the cache is *pushed* by the UAC, instead of being *pulled* by the recipient proxy, which means the attacks addressed above are prevented. We suggest that the requests with certificates (Step 6 and 8) should be transmitted over TCP. The reason will be explained in this section later.

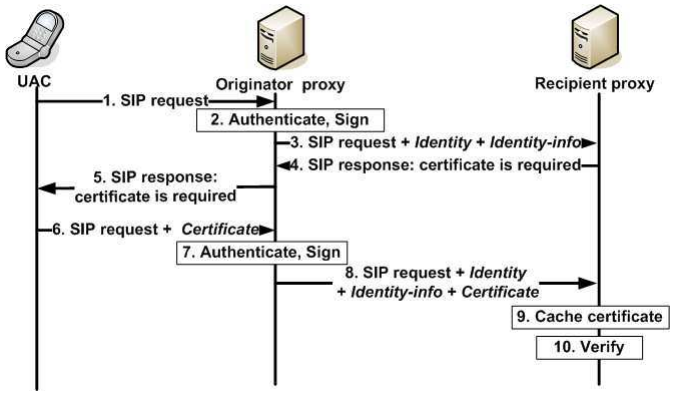


Fig. 7. An improved certificate distribution scheme

Further discussion The certificate "pushing" scheme still faces different challenges. First, in order to be transmitted properly the certificate needs to be encoded in base64. Then we have to deal with the maximum package size for each SIP package: while the maximum size of a UDP packet is 64 Kbytes including the IP and UDP headers, the maximum transmission unit (MTU) depends on the kind of network we are using but we can assume that above 1500 bytes in an ethernet network we will have to deal with packet fragmentation. SIP allows both TCP and UDP transmission protocols. In case we are using TCP for the transmission we can count on that the network level will re-assemble the fragmented SIP packages. If we are using UDP, the reception of the fragmented packet may encounter some errors.

The size of a SIP message is usually around 1000 bytes, then the size of the certificate cannot exceed 500 bytes if we want to send it over ethernet using UDP. With this limitation, we can look at RSA type certificates with a public

key of 2048 bytes and take the PEM encoding of it which is already base64. Such certificates will have an approximate size of 1350 bytes which is too much for a UDP transmission. If we take a look at Elliptic Curve Cryptography (ECC) [4] using for example a curve of type secp256k1, the approximate length of the PEM base64 encoded certificate will be around 782 bytes, which also makes it not suitable for transmission over UDP. Regarding RSA and ECC certificates, NIST recommends RSA keys of 3072 bytes for security beyond the year 2030, whilst in the case of ECC to achieve the same level of security requires a key length of 224 bytes [11].

As proposed, the infrastructure would need a caching function for the certificates and the first time they are sent it must be done over TCP. On the other hand, the idea of including certificates in the SIP messages could also be re-used to distribute the Certificate Revocation Lists (CRL) for the revoked certificates of SIP proxy servers. The CRL distribution cost would be lowered by using SIP headers to transport certificates over TCP. On the other hand, in order to support different certificate types, the RFC4474 will need to be extended to signal which kind of signature algorithm is used to include the ECC certificates which use the ECDSA signature algorithm. The current standard only specifies that all implementations must support the RSA-SHA1 signature algorithm.

5 Related work

Using reflectors for distributed denial-of-service attacks is not a new concept. Paxson [9] summarized several types of attacks using reflectors. In his paper, a reflector is defined as "any IP host that will return a packet if sent a packet." It can be a DNS server, a web server or a IP router, etc. Thus, attackers need to collect a large number of reflectors (e.g., 1 million) and send to the reflectors spoofed requests announced coming from a victim. The reflectors will in turn generate responses from themselves to the victim, so that the resource of the victim (e.g., bandwidth) is occupied. Different from these classical reflecting attacks, the attack proposed in this paper takes SIP proxies as reflectors. The attack can be successfully mounted with only a small amount of reflectors.

Our previous work [16] and [15] described some other problems and vulnerabilities of this mechanism. [16] addressed the confidentiality issues of certificate cache, which is especially vulnerable to timing attacks. In this way, attackers can observe and compare the timing required for different requests to find out the calling history between SIP domains. [15] proposed a threat by exploiting a long time for certificate downloading to block a victim SIP proxy.

6 Conclusions

With an inappropriate design of the certificates distribution scheme, the inter-domain authentication mechanism for SIP is vulnerable to Denial of Service attacks. An attacker can consume the resource of a HTTPS server by simply

selecting a list of SIP proxies as reflectors. The result of our experiments demonstrated that a HTTPS web server can be easily taken down in this way. Finally, we proposed an improved certificate distribution scheme to prevent this vulnerability. More evaluation of the proposed scheme will be done in the future.

References

1. Apache, HTTPd server. `http://httpd.apache.org/`, visited at 16th-Aug-2009.
2. T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic syntax, 2005. RFC 3986.
3. T. Berners-Lee, L. Masinter, and M. McCahill. Uniform Resource Locators (URL), 1994. RFC 1738.
4. S. Blake-Wilson, D. Brown, and P. Lambert. Use of Elliptic Curve Cryptography (ECC) algorithms , 2002. RFC3278.
5. T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2, 2008. RFC 5246.
6. Httperf. `http://www.hpl.hp.com/research/linux/httperf/`, visited at 16th-Aug-2009.
7. C. Jennings, J. Peterson, and M. Watson. Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks, 2002. RFC 3325.
8. S. Kent and R. Atkinson. Security Architecture for the Internet Protocol, 1998. RFC 2401.
9. V. Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *SIGCOMM Comput. Commun. Rev.*, 31(3):38–47, 2001.
10. J. Peterson and C. Jennings. Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP), 2006. RFC 4474.
11. Y. Rebahi, J.J. Pallares, T. M. Nguyen, S. Ehlert, G. Kovacs, and D. Sisalem. Performance analysis of identity management in the Session Initiation Protocol (SIP). In *Proceedings of the 2008 IEEE/ACS International Conference on Computer Systems and Applications*, pages 711–717. IEEE, 2008.
12. J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol, 2002. RFC 3261.
13. SIP Express Router 2.0. `http://www.iptel.org`, visited at 16th-Sep-2008.
14. SIPp. `http://sipp.sourceforge.net/`, visited at 16th-Sep-2008.
15. G. Zhang, S. Fischer-Hübner, and S. Ehlert. Blocking attacks on SIP VoIP proxies caused by external processing. *Springer Telecommunication Systems*, 2009.
16. G. Zhang, S. Fischer-Hübner, L. A. Martucci, and S. Ehlert. Revealing the calling history of SIP VoIP systems by timing attacks. In *Proceedings of the* 4^{th} *International Conference on Availability, Reliability and Security (ARES '09)*, pages 135–142, Fukuoka, Japan, 2009. IEEE Computer Society.