2009 International Conference on Availability, Reliability and Security

Revealing the Calling History of SIP VoIP Systems by Timing Attacks

Ge Zhang
Karlstad University, Karlstad, Sweden
ge.zhang@kau.se

Simone Fischer-Hübner
Karlstad University, Karlstad, Sweden
simone.fischer-huebner@kau.se

Leonardo A. Martucci
Karlstad University, Karlstad, Sweden
leonardo.martucci@kau.se

Sven Ehlert
Fraunhofer FOKUS, Berlin, Germany
sven.ehlert@fokus.fraunhofer.de

Abstract

Many emergent security threats which did not exist in the traditional telephony network are introduced in SIP VoIP services. To provide high-level security assurance to SIP VoIP services, an inter-domain authentication mechanism is defined in RFC 4474. However, this mechanism introduces another vulnerability: a timing attack which can be used for effectively revealing the calling history of a group of VoIP users. The idea here is to exploit the certificate cache mechanisms supported by SIP VoIP infrastructures, in which the certificate from a caller's domain will be cached by the callee's proxy to accelerate subsequent requests. Therefore, SIP processing time varies depending whether the two domains had been into contact beforehand or not. The attacker can thus profile the calling history of a SIP domain by sending probing requests and observing the time required for processing. The result of our experiments demonstrates that this attack can be easily launched. We also discuss countermeasures to prevent such attacks.

1. Introduction

Voice over IP (VoIP) is an innovative technology which enables its users to place calls via Internet-based infrastructures instead of traditional Public Switched Telephone Networks (PSTN). In recent years, VoIP has attracted considerable commercial interest. One of the important reasons is its low-cost. By allowing voice to be converted into packets and transported over packet-switched networks, the consolidated network serves as a foundation for enterprises to reduce operating expenses. It is predictable that in the near future, many enterprises will deploy their own VoIP infrastructures instead of using traditional PSTN equipment. The Session Initiation Protocol (SIP) [15] has been widely adopted as the signaling protocol to handle VoIP services. Similarly to the email addressing scheme, SIP VoIP users are logically grouped by means of domains. Each domain provides telephony services to its own SIP users independently. The domains also cooperate for inter-domain telephony services.

Contrary to VoIP, security threats are considered minimal in PSTN. This is achieved by not only using a closed networking environment, but also independent protocols and infrastructures. In contrast, launching an attack on VoIP is much simpler because VoIP services are based on an open environment shared with other existing Internet-based services. Therefore, VoIP can suffer from similar security threats as any other Internet services. In order to provide a trusted inter-domain context, an inter-domain authentication mechanism was proposed in RFC 4474 [13].

In this paper we present a timing attack aiming to disclose the calling history of a SIP domain. The disclosed information is not the calling history of a single user, but of a whole SIP domain. **The calling history** in this paper refers to whether a domain contacted another specific domain recently. This calling history is frequently regarded as confidential business information since other information can be predicted from it [4, 11, 10]. If such a domain is bound to a company, an attacker can aggregate the calling history during a period of time of the company. The calling history might be especially useful to predict future actions within the company. It can also reveal the company's customer base.

This vulnerability is caused by the certificate cache introduced in the inter-domain authentication mechanism. The proxy cache stores downloaded certificates to speed up the processing of subsequent SIP requests. Therefore, the time required for processing a SIP request varies whether the corresponding certificate has already been cached or not. Thus, this timing attack is based on two factors: first, the certificate of the caller's domain will be cached by the callee's proxy for each call; Second, for further requests coming from the same domain, processing will take less time since there is no need for the callee's proxy to download the cer-

978-0-7695-3564-7/09 $25.00 © 2009 IEEE
DOI 10.1109/ARES.2009.129

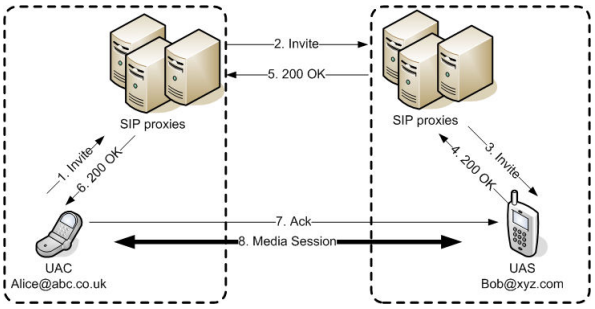

Figure 1. An overview of essential components on SIP infrastructures

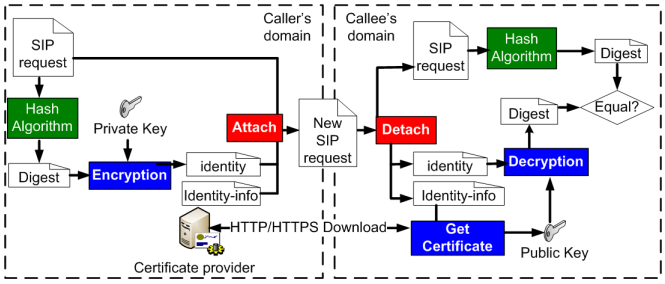

Figure 2. The mechanism of identity management as defined in RFC 4474

tificate again. In this way, an attacker can generate crafted probing requests to the victim proxy and retrieve the calling history of a domain by observing processing time. The result of our experiments show that this attack is easy to launch. To defend against this timing attack, we propose a special delay solution to uniformize the processing time which affects legal users only by a minimum degree.

This paper is organized as follows. Section 2 provides information on SIP and its mechanism for identity management. In section 3, we describe in detail how and why the timing attack works in the SIP VoIP context and we present experimental results that demonstrate the severeness of this threat. Several solutions to counteract such a timing attack will be proposed in section 4. Section 5 shows an overview of related work. Finally, we provide conclusions in section 6.

2. Voice over IP using SIP

SIP [15] works as a signaling protocol at the application layer of the TCP/IP model. It establishes or terminates media sessions between two parties. A general SIP infrastructure consists of User Agents (UA), and several SIP servers such as registrar servers, redirect servers and SIP proxies. Here, UA and SIP proxy are more related to the scope of this paper. A UA generates, terminates, or accepts SIP requests and responses. SIP proxies forward SIP requests and responses in the SIP network. SIP users are indicated by means of Uniform Resource Identifiers (URI), consisting of a pair of user name and domain name, (e.g., sip:alice@abc.co.uk), which is similar to the email format. Therefore, SIP users logically belong to different domains, and there are one or more SIP proxies in charge of processing SIP requests for each domain.

The general SIP-based telephony calling setup is shown in Figure 1. There are two SIP users in this scenario: Alice, a caller, is located in the domain abc.co.uk and Bob, a callee, is located in the domain xyz.com. First, Alice sends an INVITE request to its local proxies at (abc.co.uk). Then, the local proxies at (abc.co.uk) will forward this INVITE request to the remote proxies at (xyz.com). Then, the INVITE request will be delivered to Bob. If Bob would like to accept the request from Alice, he will reply with a 200 OK message back through the proxies. After Alice sends an ACK message to confirm the request, the signaling handshaking is accomplished. Finally, Alice and Bob will build a media session independently without the assistant of proxies, so that they can talk in this session.

Inter-domain Authentication in SIP: Many emerging attacks and spams towards VoIP have been reported recently [9, 3, 6]. Attackers and spammers frequently spoof identities in order to be untraceable. Therefore, an overall authentication mechanism is needed to identify callers. Generally, SIP proxies identify their users in their home domain. However, historically, there was no mechanism to identify callers in an inter-domain context. If the caller and callee are not in the same domain, the callee's proxy cannot guarantee that the received request is really sent from that domain as announced.

A solution based on certificates for originator authentication is proposed in RFC 4474 [13]. The purpose is to assist the SIP proxies to authenticate the source of inter-domain SIP requests. As a consequence, forging the source of a SIP request is becoming more difficult. The method is shown in Figure 2 and described as follows.

For each outgoing request to other domains, the SIP proxy in the caller's domain generates a hash digest covering several header fields (e.g., *To*, *From*, etc) and payload of this request. The digest is signed by the caller's proxy with its private key. The signature is encoded in a new header field *Identity* and included into the original SIP request. Furthermore, the SIP proxy attaches another new header field *Identity-info*, which contains the URL where the certificate can be fetched from. Then, this SIP request will be forwarded to the callee's domain.

For each incoming request from other domains, the SIP proxy in the callee's domain will first download the certificate according to the URL within the *Identity-info* field. After that, a hash digest of the incoming request will be re-

computed by the proxy. The public key retrieved from the certificate will be used to decrypt the signature contained in the *Identity* field. Finally, the result will be used to compare both hash digests. The verification is successful only if the two values are equal, otherwise, the verification fails and the proxy will reply with a response indicating a failed verification.

A domain's certificate can be used to verify all requests from this domain. Since VoIP is a real-time service, repeatedly downloading certificates for each incoming request will have a negative impact on performance. Thus, a certificate cache mechanism is recommended in RFC 4474 [13]. In this way, the downloaded certificate will be cached in the local proxy for a while. Therefore, for the next incoming request originated from the same domain, the proxy will extract the certificate from the local cache instead of downloading it again.

3. Timing Attack

3.1. Threat model

The objective of the timing attack is to retrieve confidential information by exploiting a protocol design flaw. The timing attack described in this paper reveals whether someone from a domain had been in contact recently with someone from another domain. The disclosed information is not the exact identities of caller and callee, but their domains. The attacker can be an outsider who does not belong to any domain and does not need to authenticate himself on any domain. The attacker just needs Internet access to send packets to SIP proxies. Although this kind of attack can be launched against any SIP domain which is deployed according to RFC 3261 and RFC 4474, it is most attractive against enterprises' SIP domains. Generally, two enterprises frequently communicate when they have a business connection. Therefore, the calling history might reveal such potential business connections. Companies usually have an interest to keep their business relationship confidential. In particular, companies that secretly talk about mergings would like to keep this confidential in regard to the outside world. Therefore, the disclosure of the calling history might cause direct or indirect economic losses for a company.

In the traditional PSTN context, the calling history of an enterprise is kept secret by the telephony operator, a trusted third party in a closed context. It needs more efforts for an external attacker to gather this information. However, in the SIP VoIP context, the telephony service is integrated with other ICT-based infrastructures based on an open environment, which simplifies the attack. The details of how to launch a timing attack will be shown in the following sections.

3.2. Attacking method

The inter-domain authentication defined in RFC 4474 is recommended to be deployed in future SIP VoIP services to prevent spam [14]. However, this mechanism causes other vulnerabilities. As we mentioned before, for each incoming inter-domain request, the SIP proxy should verify whether the request is really from the domain as it was announced in the request. To get the public key for verification, the callee's SIP proxy has to download the certificate according to the URL in the field of the *Identity-info* header, either through an HTTP or an HTTPS connection. Since the downloaded domain certificate is valid to authenticate all the requests from the same domain, there is no need to redownload the certificate again for further requests from the same domain. The downloaded certificate will be retained in the certificate cache at the callee's proxy for a while in favor of the next request. In this way, the content of certificate cache can reveal the calling history in form of which domains contacted this domain recently. For example, if domain A contacted domain B recently, the certificate of domain A should be found in the certificate cache at domain B's proxy. Therefore, the attacker can infer the calling history of domain A when he knows whose certificates are cached at domain A's proxy.

We introduce a timing attack to reveal the content of the certificate cache. As mentioned before, the processing time for each request varies, depending on whether the certificate of a domain is cached by the victim proxy or not. If the certificate of the domain has already been stored in the local cache, the SIP proxy can use it without downloading it again. Then, the total processing time for such a request is relatively short. Otherwise, the victim proxy has to download the certificate, which takes more time.

Therefore, the attacker first selects a victim domain as the attacking target. He also selects some domains that might have connections with the victim domain. We call these domains **testing domains**. Next, he can deliberately generate fake requests which seem to come from the testing domains, and then send these requests to the proxy in the victim domain. We call these requests **probing requests**. When the victim proxy receives the probing requests, it will verify them to see whether they are really sent from a testing domain. Therefore, the victim proxy will verify the requests using the certificates of testing domains, either extracting from the local cache or downloading from the remote certificate provider. Since the attacker does not own the private key of the testing domains, he cannot generate the correct signature. Hence, the verification will fail, and the victim proxy will reply with a verification failed response to the attacker. To ensure that such a response will be delivered back to the attacker, and not to the testing domains, the attacker can simply encode his IP address in the first *Via* field[1]. In

[1] A *Via* header field indicates the location where the response is to be

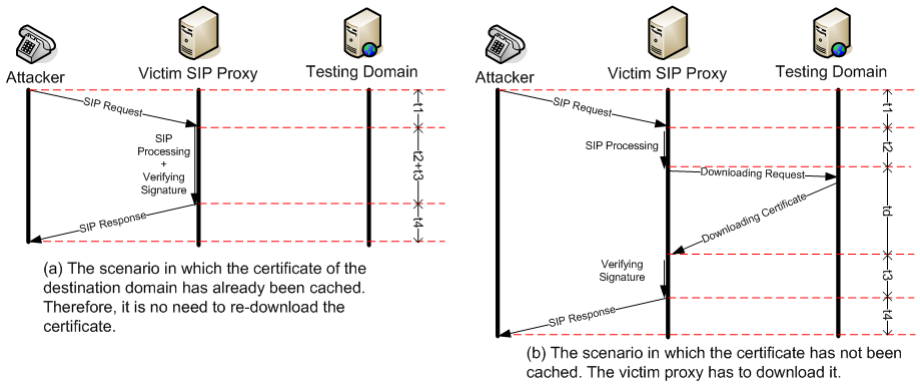

Figure 3. The working flow shows the comparison of time cost in two situations, one with the certificate already cached, another not

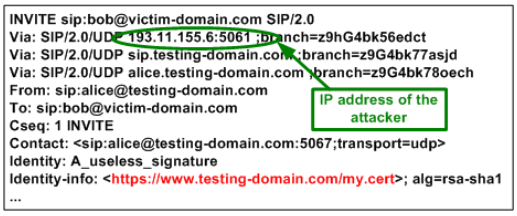

Figure 4. A sample probing request

this way, the attacker is able to find out whether the testing domains contacted the victim domains recently by comparing the time interval between the request and the response.

For example, given a victim domain `victim-domain.com` and a testing domain `testing-domain.com`. If an attacker tries to find out whether these two domains had contacted each other recently, he can send a probing request to the SIP proxy in `victim-domain.com` like Figure 4, with the *Identity-info* filled with the URL of `testing-domain`'s certificate. And the first *Via* field is encoded with the IP address of the attacker. The signature in the *Identity* field is randomly created. Nevertheless, the attacker does not care whether the request can be successfully authenticated, instead, he is just concerned about the response time of this request.

How the victim proxy handles the probing requests in detail is shown in Figure 3. We define:

t_{sum}: The total time for processing a probing request.

t_1: The time for the probing request transmitting from the attacker to the victim proxy.

sent. In some cases, a SIP request has to be passed through several intermediate hops between two domains and the IP addresses of the intermediate hops will be recorded in the *Via* field in favor of routing back the corresponding response. Therefore, it is natural if the IP address in the *Via* field does not belong to the caller's domain.

t_2: The time for the request pre-processing, including message parsing, etc.

t_3: The time for the verification of the caller's identity.

t_4: The time for the response transmitting from the victim proxy to the attacker.

t_d: The time for downloading the certificate through an HTTP or an HTTPS connection.

In Figure 3(a), the attacker first sends a probing request to the victim proxy. Secondly, the victim proxy will preprocess the message. Then, since the related certificate has already been cached in the victim proxy, there is no need to download it. Thus, the victim proxy will use the cached certificate to verify the request directly. The verification should fail, and the victim proxy will reply with the failure reason to the attacker. The total time is

$$t_{sum} = t_1 + t_2 + t_3 + t_4$$

The procedure shown in Figure 3(b) is similar to the one in Figure 3(a). The only difference is that the victim proxy has to download the certificate remotely since the required certificate has not been cached yet. The total time for processing the request is

$$t_{sum} = t_1 + t_2 + t_d + t_3 + t_4$$

The proxy has to spend an additional t_d time on processing a request if the corresponding certificate is not cached by the victim proxy.

3.3. Testbed Setup

To find out whether t_d is large enough to distinguish between requests whose certificates are cached or not cached, we performed a series of experiments based on a testbed,

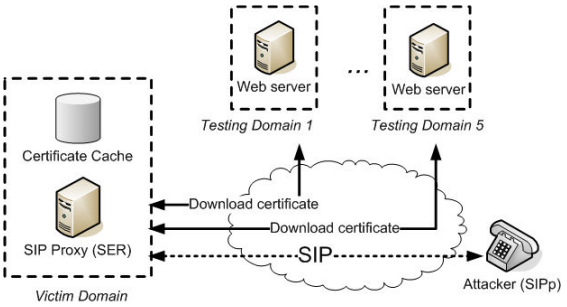

Figure 5. Testbed architecture

whose architecture is illustrated in Figure 5. It consists of the following 3 components:

- A victim SIP proxy as the attacking target. The proxy is implemented according to the mechanisms defined in RFC 3261 and RFC 4474. We employ SIP Express Router (SER) [1] for this task. The proxy is equipped with a certificate cache to store downloaded certificates.

- External certificate providers. We selected five certificate providers from the internet. Three of them support certificate downloading through both HTTP and HTTPS connections. One only supports HTTP downloading and another only supports HTTPS downloading. Each certificate provider has its own domain name. They simulate the **testing domains** we introduced in the previous section. Readers should notice that these domains do not provide SIP services and they are not real SIP domains. They only provide certificates.

- An attacking tool. The attacking tool is able to generate probing requests which are forged as coming from the testing domains. The *Identity-info* header field contains the URL of the certificate from a provider. Since the attacker does not know and has no need to know the private key of the testing domains in this attacking scenario, the *Identity* header field is simply filled with a random string. The attacking tool sends the probing requests to the victim proxy and is waiting for the response. As soon as it receives a response from the victim SIP proxy, the attacking tool records the processing time t_{sum}. The attacking tool is implemented by using SIPp [2], an open source SIP message generating tool.

The testbed of the victim domain (SIP proxy) is established on a Pentium 4 machine with 512 MB RAM running Linux Ubuntu operating system and Internet access. The attacking tool is running on another machine with the same configuration.

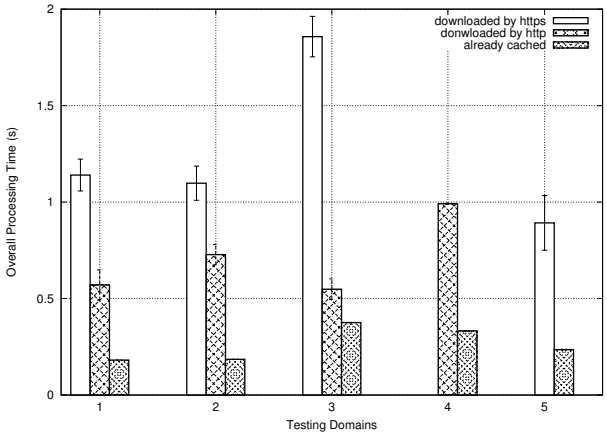

Figure 6. Distribution of SIP request response times for three different setups

3.4. Testing and Testing Result

In the experiments, we used the attacking tool to send the probing requests to the victim proxy in three setups as follows.

- **The certificates have already been cached in the victim proxy, no downloading needed.** Before the experiments, the attacking tool sends the probing request to the victim proxy in order to let the victim proxy cache the certificates. As a consequence, the victim proxy downloads the certificates and caches them locally. Then, we start our experiments.

- **The certificates are not cached and the victim proxy has to download the certificate by HTTP connection.** To assure no certificate is cached, we just simply restart the victim SIP proxy to clear up the cache. Besides we set the option of the attacking tool to send probing requests including *Identity-info* starting with **http://**. As a consequence, the victim proxy will download the certificates through an HTTP connection.

- **The certificates are not cached and the victim proxy has to download the certificate by HTTPS connection.** The same as the previous setup, the certificate cache should be cleared up by restarting the victim proxy. During the experiments, the probing requests generated by the attacking tool include *Identity-info* starting with **https://**. Therefore, the victim proxy will download the certificates through an HTTPS connection.

We sent the probing requests to the victim proxy in the three setups and measured the time interval between sending each request and receiving the response. This time interval is the overall processing time t_{sum} introduced in the

previous section. We repeated each test ten times and the result is shown in Figure 6. The figure shows the mean and standard deviation of t_{sum} for the probing requests. Readers should bear in mind that the domain of example 4 does not support HTTPS and the domain of example 5 does not support HTTP.

The data clearly shows the difference between the cases where the certificate is already in the cache or not. The gap even increases when the victim proxy downloads the certificate by HTTPS, since the handshake of HTTPS takes more time. From the measurements, we found that t_{sum} is less than 0.5 s with little deviation if its domain certificate is already cached. Otherwise, t_{sum} varies between 0.5 and 1 s if the certificate is downloaded through an HTTP connection. And it takes 1 to 2 seconds if the certificate is downloaded through an HTTPS connection. Furthermore, the standard deviation of t_{sum} can be observed clearly from the figure if the certificate is not cached. The variance is caused by the variable network conditions in downloading. In contrast, without downloading, the standard deviation of t_{sum} is too small (around 0.4 ms) to be observed from the figure if the certificate has been cached.

Hence, the attacker can repeatedly send the same probing request to the victim proxy several times during a period and observe t_{sum}. We call the processing time for the first probing request t_{sum1}. Whether the victim proxy will download the certificate for the first probing request is unknown and is what the attacker wants to find out. However, it is obvious that the proxy does not need to download it for the subsequent probing requests because the certificate should be in cache already, either by the first probing request or even cached earlier. Then, the attacker can calculate the mean ($mean$) and standard deviation ($stdev$) of subsequent t_{sum} as the benchmark, which is used to be compared with t_{sum1}. t_{sum1} is assumed to be including a downloading time t_d if it falls outside a *confidence range*:

$$mean \pm d \times stdev$$

d can be arbitrarily selected by attackers. According to Chedyshev's inequality [16], the probability of false negative is at most $1/d^2$. Then, the attacker can know whether such a certificate is cached or not by comparing t_{sum1} with the benchmark and confidence range. In this way, the attacker can profile the list of the certificate cache of any SIP domain which supports inter-domain authentication mechanism and get information about their calling history.

4. Countermeasures

Since the probing requests for timing attack are all well-formed SIP messages, the attack is difficult to be detected by using techniques based on message fingerprint. Furthermore, the attack works with only a small amount of probing requests, so it unlikely leads to traffic anomalies. Therefore, regular Intrusion Detection Systems (IDS) for SIP cannot be effective to provide a countermeasure against this kind of attack. In this section, we analyze other countermeasures for this attack.

Solution 1: One possible solution is to avoid using a certificate cache. Without cache, this timing attack cannot succeed. However, the performance of the SIP infrastructures will decrease significantly without the assistant of caches. Further, without cache, the SIP proxies become ideal reflectors [12] which can be exploited to launch Distributed Denial of Service attacks to the certificate server.

Solution 2: Another solution is to ensure that the SIP proxy always takes a constant amount of time to process an incoming SIP request. If we eliminate the gap of the time difference between requests whose certificates are cached or not cached, the timing attack will not work anymore. However, we cannot accelerate the processing for requests whose certificates are not in cache. Therefore, the only option would be to delay responses on purpose, to ensure that all t_{sum} are equal. Different to Solution 1, the cache mechanism still works in this solution, so there is no need to repeatedly download the same certificate. Thus, there is no additional overhead of bandwidth caused. However, the legal SIP users will still encounter an additional delay. For example, as can be seen in Figure 6, it takes up to 2 seconds for a SIP proxy to download a certificate through an HTTPS connection. Therefore, the SIP proxy will delay the response for 2 seconds on purpose. As a result, the legal users have to spend additional 2 seconds on each calling request.

Solution 3: Since attackers do not know the private keys of the testing domains which they counterfeit in the probing requests, all their probing requests sent to the victim proxy will not be verified successfully. Therefore, the proxy can refuse to make any response for unverified requests. In this way, the attacker can neither get the response nor can he observe the processing time of the probing requests. Therefore, this kind of attack does not work. Since the requests from the legal users should be correctly signed by their proxies, this solution is unlikely to affect the usage of legal users. However, RFC 4474 states that the responses MUST be returned [13]. It might be due to the consideration of system robustness. Thus, this solution would not be compliant to RFC 4474.

Solution 4: A better solution is a combination of solutions 2 and 3. In this way, the SIP proxy does not delay all the responses. Instead, it only delays the responses to unverified requests. Contrasty, for verified requests, the SIP proxy will send the responses immediately without any delay. We set a constant t_{delay} and the mechanism ensures that $t_{sum} \equiv t_{delay}$ for each unverified request. The t_{delay} should be set equal to or higher than the possible maximum t_{sum}. The solution consists of two algorithms, Algorithm 1

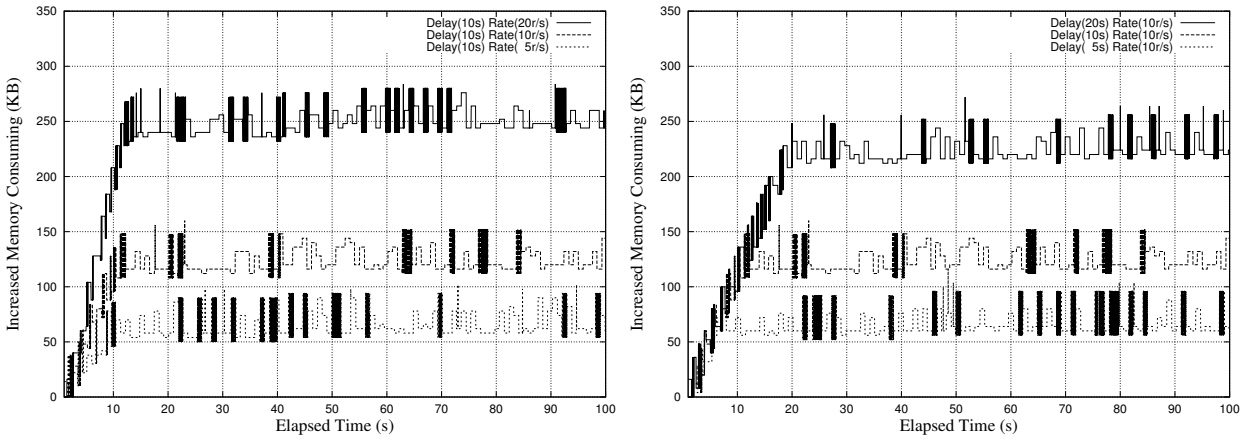

Figure 7. The consumed memory increases by introducing Solution 4

and 2. In Algorithm 1, the proxy has two sets, one called T used to store the time stamp for each SIP request when the proxy receives it, another called R used to store the responses which need to be delayed. Shown as Algorithm 1, both T and R are first empty. When the proxy receives an incoming SIP request, the request will be assigned a unique identifier. Then, the proxy generates a current time stamp for this request and inserts the time stamp into set T. Next, the proxy will continue to process this request and generate a response. If such a response is not due to failed verification, the proxy sends the response to the user directly and removes the corresponding timestamp stored in T. Otherwise, the proxy puts this response into set R to delay it. The SIP proxy executes Algorithm 2 repeatedly every period of intervals, aiming to send the responses which are already delayed enough time in the R. The proxy goes through all the elements in the set R with checking their timestamp in the set T. If the difference between current time and the timestamp is more than t_{delay}, which means such a response has been delayed enough time, the proxy will send it out and remove it from R. Otherwise, the proxy will skip this request and continue to check the next one.

Solution 4 has advantages over the previous ones. Its affect on legal users is minimized since their requests should contain correct signatures, which means that their responses should not be delayed. The proxy only delays the responses to those requests which do not contain valid signatures. In this way, the attacker cannot profile the content of the certificate cache for the SIP proxy by timing attack because t_{sum} for all the probing requests is almost constant. However, this method introduces a processing overhead, especially higher memory consumption. Thus, an attacker can flood the proxy with probing requests to cause a Denial of Service.

To investigate this problem, we implemented a prototype of solution 4 using a linked-list to save the delaying responses. When a delaying response is generated, it will be inserted into the list (here some memory resource will be allocated). After timeout of t_{delay}, the response will be sent and removed from the list (here the occupied memory resource will be freed).

Algorithm 1: Classification

```
    Input: T, R
1.1 T = ∅; R = ∅; i = 0 ;
1.2 foreach m_i = GetSipRequest() do
1.3     t_i = GetCurrentTime();
1.4     T = {t_i} ∪ T;
1.5     r_i = SipProcess(m_i);
1.6     if IsUnverified(r_i) == TRUE then
1.7         R = {r_i} ∪ R;
1.8     end
1.9     else
1.10        SendSipResponse(r_i);
1.11        T = T\{t_i};
1.12    end
1.13    i++;
1.14 end
```

We performed experiments to monitor the memory consumption after deploying solution 4. We set t_{delay} from 5 to 20 seconds on the proxy, and we set the attacking rate from 5 probing requests per second to 20 probing requests per second on the attacking tool. As can be seen from Figure 7, the memory consumption increases with the growth of t_{delay} and attacking rate. In the figure, the maximum memory usage is around 270 KB, but it can raise if the attacker sends the probing requests faster. Therefore, we suggest that the t_{delay} should be confine in a reasonable range and this delay solution should also be used combining with other general anti-flooding mechanisms, e.g. [7].

Algorithm 2: Ddelay

Input: T, R, t_{dalay}

2.1 $i = 0$;
2.2 $t_{now} =$ `GetCurrentTime()`;
2.3 **foreach** $r_i =$ `GetSipResponse(`*R,i*`)` **do**
2.4 $t_i =$ `GetTimeStamp(`*T,i*`)`;
2.5 **if** $t_{now} - t_i \geq t_{delay}$ **then**
2.6 `SendSipResponse(`r_i`)`;
2.7 $R = R \backslash \{r_i\}; T = T \backslash \{t_i\}$;
2.8 **end**
2.9 $i++$;
2.10 **end**

5. Related Work

Felten and Schneider [8] presented a timing attack aiming to compromise the privacy of users web-browsing histories. Their method works by exploiting the fact that once a web user visits a website, some related information or files of the website will be cached on the local machine to speed up the subsequent visits. Therefore, a malicious website can detect whether the user visited a given webpage recently by comparing the time required by certain operations. A. Bortz, et al. [5], extended the timing attack to two types: direct timing and cross-site timing. The direct timing attacks directly measure the response time for several HTTP requests aiming to expose information such as validity of a username. The cross-site timing attacks are launched by a malicious website aiming to profile the users current status on anther website. Different from their scenarios, the timing attack presented in this paper focuses on the SIP VoIP system. Furthermore, compared with the web scenario, the time differences are more significant and thus easier to be observed in our attacking scenario on VoIP. Finally, the countermeasure that we propose differs from the previous solutions. We can roughly distinguish legal users and attackers in VoIP scenario and our delay solution should not affect the usage of legal users.

6. Conclusion and future work

This work described how timing attacks on SIP VoIP infrastructures can potentially reveal the calling history of a VoIP domain. Such attacks may be exploited by business espionage, as the calling history is useful to profile business connections of companies. Our experiments also show that the attack can be easily launched by using open source software. Furthermore, we discussed several solutions to prevent the attack, and proposed a combined solution which affects less to legal SIP users. In the future, we will investigate more potential vulnerabilities on SIP VoIP infrastructures caused by a cache, such as DNS cache.

References

[1] SER. http://www.iptel.org, visited at 16th-Sep-2008.
[2] SIPp. http://sipp.sourceforge.net/, visited at 16th-Sep-2008.
[3] VoIP SA. http://www.voipsa.org/, visited at 16th-Sep-2008.
[4] Who has access to my phone conversation and text message records? http://www.privacyalerts.org/phone-records.html, visited at 16th-Sep-2008.
[5] A. Bortz and D. Boneh. Exposing private information by timing web applications. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 621–628, New York, NY, USA, 2007. ACM.
[6] V. G. Cerf. Spam, spim, and spit. *Commun. ACM*, 48(4):39–43, 2005.
[7] S. Ehlert, G. Zhang, D. Geneiatakis, G. Kambourakis, T. Dagiuklas, J. Markl, and D. Sisalem. Two Layer Denial of Service Prevention on SIP VoIP Infrastructures. *Computer Communications*, 31(10):2443–2456, June 2008.
[8] E. W. Felten and M. A. Schneider. Timing attacks on web privacy. In *CCS '00: Proceedings of the 7th ACM conference on Computer and communications security*, pages 25–32, New York, NY, USA, 2000. ACM.
[9] D. Geneiatakis, T. Dagiuklas, G. Kambourakis, C. Lambrinoudakis, S. Gritzalis, S. Ehlert, and D. Sisalem. Survey of security vulnerabilities in session initiation protocol. *Communications Surveys & Tutorials, IEEE*, 8(3):68–81, 2006.
[10] S. Hinde. Call record analysis. In *IEE Colloquium '96: Proceedings of the Making Life Easier-Network Design and Management Tools*, pages 8:1–8:4, 1996.
[11] A. P. Macarthur. The NSA Phone Call Database: the problematic acquisition and mining of call records in the United States, Canada, the United Kingdom and Australia. *Duke Journal of Comparative and International Law*, 17(2):441–482, 2007.
[12] V. Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *SIGCOMM Comput. Commun. Rev.*, 31(3):38–47, 2001.
[13] J. Peterson and C. Jennings. Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP), 2006. RFC 4474.
[14] J. Rosenberg and C. Jennings. The Session Initiation Protocol (SIP) and Spam, 2008. RFC 5039.
[15] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol, 2002. RFC 3261.
[16] S. M. Ross. *Introduction to probability and statistics for engineers and scientists*. John Wiley & Sons, 1st edition, 1987.