Master Thesis
Computer Science
Thesis no:
March, 2007

Denial of Service on SIP VoIP Infrastructures Using DNS Flooding

- Attack Scenario and Countermeasures –

Department of
Security Engineering
School of Engineering
Blekinge Institute of Technology
Box 520
SE-372 25 Ronneby
Sweden

This thesis is submitted to the department of Security Engineering, School of engineering at Blekinge Institute of Technology, in partial fulfillment of the requirement for the degree of Master of Science in Computer Science. This thesis is equivalent to 16 weeks of full time studies.

Contact information:
Author:
Ge Zhang
Email: nickchang918 -AT- hotmail.com

External Advisor:
Mr. Sven Ehlert
Address: Fraunhofer Institute FOKUS, Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany
Telephone: +493034637378
Email: ehlert -AT- fokus.fraunhofer.de

University Advisor:
Dr. Bengt Carlsson
Department of Security Engineering
Telephone: +46457385813
Email: bca -AT- bth.se

Department of
Security Engineering
School of Engineering
Blekinge Institute of Technology
Box 520
SE-372 25 Ronneby
Sweden

Acknowledgement

This work has been partly conducted in the European Union funded IST-COOP-5892 project SNOCER (www.snocer.org). The paper included in this thesis was submitted to *ACM IPTCOMM conference 2007*. I gratefully acknowledge the research opportunity offered by Fraunhofer FOKUS Institute and especially thank the altruistic help from my supervisors, Mr. Sven Ehlert and Dr. Bengt Carlsson. Finally, I wish to acknowledge with gratitude the continued support and encouragement given by my parents and my love, Hui Xie.

Abstract

A simple yet effective Denial of Service (DoS) attack on SIP servers is to flood the server with requests addressed at irresolvable domain names. In this thesis we evaluate different possibilities to mitigate these effects and show that over-provisioning is not sufficient to handle such attacks. As a more effective approach we present a solution called the DNS cache solution based on the usage of a non-blocking DNS cache. Based on various measurements conducted over the Internet we investigate the efficiency of the cache solution and compare its performance with different caching replacement policies applied.

Introduction

As opposed to PSTN (Public Service Telephone Network), VoIP (Voice over Internet protocol) providers such as skype are more and more welcomed with taking advantage of its low cost. However, the new network problems, like packet loss and Quality of Services (QoS) are emerging to the VoIP users. In the past, Security threats are considered minimal in current circuit switched networks. This is achieved by using a closed networking environment dedicated to a single application. However, for the VoIP services, which are based on an open environment such as internet, the systems are totally exposed to the attackers. In order to afford a broad service, the VoIP proxies can be accessed with a flat Internet access rate by anybody. Therefore, it is possible for an attacker to launch a DoS (Denial of Service) attack to VoIP proxies with a low cost.

SIP (Session Initial Protocol) is a protocol proposed standard for initiating, modifying and terminating an interactive user session that involves multimedia elements such as video, voice. It is one of the leading signalling protocols for VoIP. Whereas, SIP depends much on DNS (Domain Name Service), and this feature could be exploited by attackers to launch a DoS attacking by difficult-resolvable DNS flooding. In my thesis work, I will investigate this attack and give a possible countermeasure.

Research questions

This research is focus on a special DoS flooding attack to SIP system. Mentioned in my proposal, I will answer three questions as following:

- How to find a proper method to mitigate the effect of DoS attack via DNS request?
- Which factors of DNS cache and SIP proxy (e.g. caching replacement policy, cache entry number, parallel processes number of proxy, etc) are useful to deal with this problem?
- Which kind of combination of the useful factors is the most efficient?

By answering these research questions, a problem and a solution to this problem will be verified.

My contributions to this paper

This kind of attacking had already been noticed by Fraunhofer FOKUS institute before my work started. But most of the work was based on theory at that time. In my research work, I did the threat evaluation of this DNS flooding attack in a simulated environment and showed that this attack slowed down message processing of SIP proxy by a fair amount. Secondly, I developed the prototype --- a DNS cache with some special functions in SIP environment by C in Linux. Finally, I performed several experiments in the simulated test bed to answer the research questions.

Outline of thesis

The thesis is organized as follow. Firstly, I will exhibit recent study outcome by one of our research papers, "*Denial of Service on SIP VoIP Infrastructures Using DNS Flooding*". Secondly, in the conclusion part, I will answer the research questions and discuss some problems in the research with possible improvements in the future. Finally, I attached some details of my work in the appendix.

Denial of Service on SIP VoIP Infrastructures Using DNS Flooding

- Attack Scenario and Countermeasures –

Ge Zhang, Sven Ehlert, Thomas Magedanz
Fraunhofer Institute FOKUS, Berlin, Germany
{zhang, ehlert, magedanz}@fokus.fraunhofer.de

Dorgham Sisalem
Tekelec, Berlin, Germany
sisalem@tekelec.com

ABSTRACT

In this paper we address the issue of a special denial of service (DoS) attack targeting a subcomponent of a Session Initiation Protocol (SIP) based VoIP network. Our focus is targeted at attacks that are addressed at the Domain Names Service (DNS). By flooding a SIP element with messages containing difficult-resolvable domain names, it is possible to block the target for a considerable amount of time. We evaluate possibilities to mitigate these effects and show that over-provisioning is not sufficient to handle such attacks. We present results gained from testing with actual SIP providers of a counter solution based on a non-blocking DNS caching solution. Within this cache we evaluate different caching strategies and show that the Least-Frequently-Used caching strategy gives best results to mitigate this kind of attack.

Categories and Subject Descriptors:

C.2.0 **[Computer-Communication Networks]**: Security and protection;

General Terms

Security, Measurement, Experimentation

Keywords

SIP, DoS, DNS

1. INTRODUCTION

Security threats are considered minimal in current circuit switched networks. This is achieved by using a closed networking environment dedicated to a single application (namely voice). In an open environment such as the Internet, mounting an attack on a telephony server is, however, much simpler. This due to the fact that VoIP services are based on standardized and open technologies (i.e. SIP or H.323) using servers reachable through the Internet, implemented in software and provided often over general purpose computing hardware [1]. A special security concern is flooding with malicious or useless messages which can waste a considerable amount of resources of the SIP server. Instead of generating a multitude of costly voice calls, the attacker can easily send thousands of VoIP invitations in a similar manner to attacks on Web servers. These attacks are simple to mount and with flat rate Internet access are inexpensive for the attacker.

Denial of Service (DoS) attacks [2][3] aim at denying or degrading a legitimate user's access to a service or network resource, or at bringing down the servers offering such services. According to a 2004 CSI/FBI survey report 17% of respondents detected DoS attacks directed against them, with the respondents indicating that DoS was the most costly cyber attack for them, even before theft of proprietary information [4].

Several possibilities exist for an attacker to cause a Denial of Service in a VoIP infrastructure [5]. Besides launching brute force attacks by generating a large number of useless VoIP calls, attackers can use certain features of the used VoIP protocol to incur higher loads at the servers. Further, the VoIP infrastructure can be corrupted by launching DoS attacks on components used by the VoIP infrastructure or the protocols and layers on top of which the VoIP infrastructure is based such as routing protocols or TCP. In this paper we investigate a special DoS attack that is launched utilizing the Domain Name Service, on which SIP heavily depends on, which we call a *SIP DNS attack.* We show that this attack is easy to launch and slows down message processing by a fair amount. We evaluate possibilities to mitigate effects of this attack and show that simply over-provisioning is

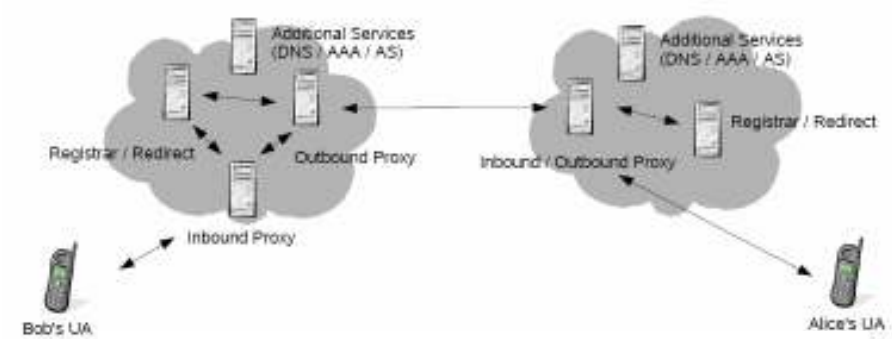


Figure 1: Essential components and their functions of a SIP infrastructure

not sufficient to counter the effects. We present a solution based on a non-blockable cache design and give results gained from testing with actual SIP providers. Within our cache solution we evaluate different caching strategies and show that the Least-Frequently-Used algorithm gives best results to mitigate the effects of this attack.

This paper is organized as follows. In Section 2 we present an overview of the SIP signalling protocol and the special usage of the Domain Name Service within SIP. In Section 3 we describe in detail the SIP DNS attack and demonstrate its effectiveness. In Section 4 we present our test bed, while in Section 5, we provide currently available solutions which can be deployed to counter this attack, and give an analysis of their feasibility and limitation. We outline our own solution and evaluate it based on our test bed in Section 6. Finally we summarize our work and suggest further steps in this research direction.

1.1. Related Work

Recently there has been an increase in VoIP security awareness, as governmental institutions are becoming aware of the situation (e.g. publications by the US National Institute of Standards and Technology [6]). In this report, the researchers classified and analysed theoretical threats to confidentiality, integrity and availability of SIP system from different aspects. Finally, they mentioned a possible "CPU resource consumption attack without any account information" in the appendix which is similar to our research.

Other works on Denial of Service Protection on SIP servers exit, however they don't focus on DNS related attacks. Geneiatakis et al. [7] propose a framework to defend against malformed SIP messages by a signature-based technique. SIP Grammar corrected will be applied to every incoming SIP messages and malformed messages will be discarded. E.Y. Chen [8] proposes a concept for detecting DoS Attacks on SIP systems using a SIP state machine model. The system is designed to detect unauthorized invalid message flooding and malformed messages, however no measurements are given in the paper. Sengar er al. [9] have devised a DoS detection mechanism based on statistical anomaly detection. In the experiment of detecting TCP SYN flooding, UDP-based RTP packets flooding and SIP-based INVITE flooding, the prototype shows high accuracy against high-rate attacks. Another online detection mechanism based on Bayesian Model for SIP is proposed by Nassar et al. [10]. The system is able to detect different kinds of threats towards VoIP applications besides DoS, including SPIT and Password cracking.

2. BACKGROUND

2.1. Session Initial Protocol

The Session Initiation Protocol (SIP) [11] is ever more establishing itself as the standard for VoIP services in the Internet and next generation networks.

A basic SIP infrastructure consists of several components (see *Figure 1*), including *User Agents* that generate or terminate SIP requests, *Registrars*, where users log in and announce their availability in the SIP network and *Proxies* that forward requests in the SIP networks. Several proxies can be deployed in a SIP infrastructure, e.g. *outbound proxies* that regulate routing outgoing traffic from one network to a foreign network and *incoming proxies* that handle all incoming SIP requests possibly enforcing additional security checks.

SIP is a text based protocol designed to establish or terminate a session between two partners. The message format is similar to the HTTP protocol [12], with message headers and corresponding values, e.g. FROM: user@sip.org to denote the sender of a message. The destination of a SIP messages (*Request-URI*) is provided in the first line of the message, the *request line*. Additionally, several other message headers are dedicated to routing purposes in the network.

2.2. Domain Names Service

The Domain Name Service (DNS) is the basis for most current internet services available today, including web and email.

It is a completely globally distributed and managed database, providing an essential service for Internet applications and users i.e. name resolution [13] [14], which is the mapping from human readable textual domain names (e.g. www.berlin.de) to a numerical IP address (e.g. 62.50.41.196). Whenever a user requests a domain resolve, there are generally two cases to distinguish:

- *The DNS server knows the name mapping*. The name server might know the mapping because it is the authoritative name server for this domain. As such, all mappings for the domain are preconfigured for this domain server. The server might also know the name because it has resolved this address previously. Generally, in this case the mapping is still stored in the server's internal cache.
- *The DNS server does not know the name mapping*. In this case the server will issue a recursive request to other name servers that might be able to provide an answer. The server will eventually receive a response, either containing the valid mapping or an error message that no mapping is possible. In the former case, the mapping will be stored in the server's internal cache for a limited period of time. The names server can also set a time limit for the query. If no answer is received within this limit, the address is considered unresolvable.

2.3. DNS Usage in SIP Infrastructures

The Domain Name Service plays a key role in every SIP network at three following aspects [15].

- Many of the header fields in a SIP message contain Fully Qualified Domain Names (FQDN) that need to be resolved for further processing from a SIP entity.
- To interconnect the Public Switched Telephone Network (PSTN) with a SIP network, ENUM telephone number mapping [16] is used. In short, this allows the mapping of a PSTN telephone number (e.g. +1 234 567) to a valid SIP number, if this mapping has been previously established using the domain name service.
- SIP can utilize different transport level protocols (e.g. UDP or TLS). To find its right contact server in regard to the used transport layer protocol, a SIP entity will issue a DNS SRV [17] request for the domain of the regarding SIP URI. The response will contain one or more destination hosts that provide the required service.

In short, a SIP entity might query the DNS subsystem up to three times (ENUM mapping, server locations and address resolution) before it can actually process and forward a message.

3. SCOPE OF THE ATTACK

The goal of a DoS attack is to render the service inoperable for as long as possible. While the kind of attack we describe here can be launched at any kind of SIP entity (user agent, proxy, registrar, and redirect). It is most effective against proxies or registrars/redirectors [2]. In the following we will refer to these possible targets as SIP servers.

Whenever a SIP server encounters a fully qualified URI in a header field necessary for routing (e.g. `VIA` or `Route` field), it issues a query to the name server to receive a valid address mapping. On average it takes 1.3 DNS queries to receive an answer with the mean resolution latency less than 100 ms [18]. However, due to configuration errors, these numbers can be considerably higher [19].

The SIP DNS attack targets this relatively high processing time. It is possible to disturb server operation with specially crafted SIP messages containing URIs that will cause an even higher processing time at the DNS server by taking into account an URI of which the attacker is sure that its mapping will not be in the cache of a name server and the URI will trigger a request to an authoritative name server that has a common low response time, (e.g. because of low bandwidth connection). The former case is easy to generate by adding random host names to the left side of the address domain. The latter case can be easily discovered by querying different name servers and measuring reply times. As an example for such a SIP message, see *Figure 2*

```
INVITE: SIP:u1@2d4fww.hard-to-resolve.domain SIP/2.0
Via: SIP/2.0/UDP 10.147.65.91; branch=z9hG4bk29FE738
CSeq: 16466 INVITE
To: sip:u1@2d4fww.hard-to-resolve.domain
Content-Type: application/sdp
From: SIP: u2@2d4fww.hard-to-resolve.domain; tag=24564
Call-ID: 1163525243@10.147.65.91
Subject: Message
Content-Length: 184
Contact: SIP: u2@2d4fww.hard-to-resolve.domain
…
<SDP part not shown>
```

Figure 2: Example SIP Message with Unresolvable URIs

Such a message is a well formatted message that complies with the SIP standard in every respect and as such cannot easily be filtered out by a SIP server or an Intrusion Detection System [7].

Issuing SIP queries with a variation of such URIs will stop operation at a SIP server for a considerable time, as the SIP server can only continue its operation after having received an answer from the DNS server. For example, the SIP server will wait up to five seconds from a BIND DNS server [20] which is commonly used to resolve a request. If it doesn't receive any answer from the BIND DNS server within five seconds, this domain name will be regarded unresolveable and the SIP server will continue to deal with the next one. The whole processing of is shown in *Figure 3*.

Thus, a SIP DNS attack can be launched easily by sending multiple messages containing unresolvable names within.

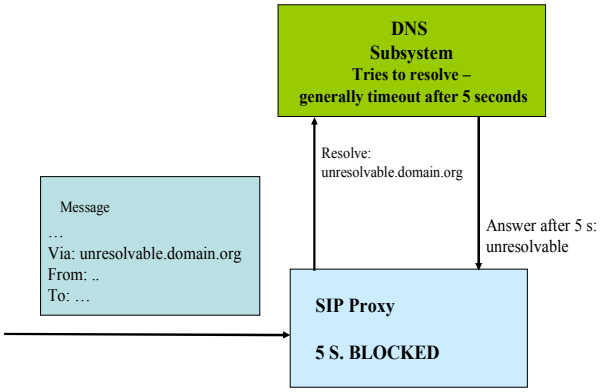


Figure 3: The Attacking Scenario by blocking SIP proxy with messages contain unresolvable URIs

4. TEST BED AND INSTRUMENT

Within our test bed we prove the effectiveness of the attack and evaluate countermeasures against it. The test bed consists of five main components.

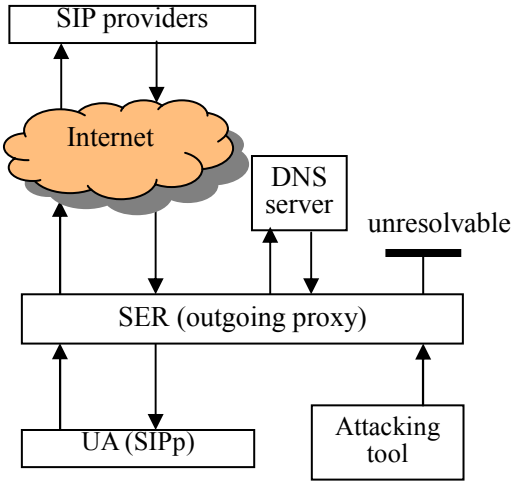


Figure 4: Test bed architecture

- A SIP proxy as the main target of the attack. In our test bed, all messages to or from a caller have to go though this proxy. We have used the SIP Express Router (SER) [21] for this task. SER is a SIP server which can act as SIP registrar, proxy or redirect server.
- A local DNS server. The SIP proxy is configured to contact this server for DNS requests.
- An attack tool generating SIP messages containing unresolvable domain names. We have developed such a tool that can continuously send random messages with different hard to resolve domain names to our proxy.
- User Agents (UA) representing legal users that register themselves on remote SIP servers. We have set them up with the SIPp message generating tool [22]. SIPp is a SIP protocol traffic generator tool and can send and reply to arbitrary SIP messages, such as INVITE, REGISTER to other entities in a specified time interval and with defined reply codes.. We use SIPp to simulate regular SIP REGISTER traffic, consisting of REGISTER requests with different kinds of responses from remote servers.
- External SIP providers. We have chosen 100 different SIP providers from all over the world, mostly located in Europe and North America. The User Agents will be registered there. Every external SIP provider is located at a different domain.

The test bed was established on Pentium D double processor machines with 1 GB RAM (Proxy, User Agent, and Attack tool) running on Linux Operating Systems, equipped with 100 M Bps internet access.

The logical structure of test bed shows is shown in *Figure 4*. We have simulated the scenario with the following steps.

- The SIP proxy is setup first and can be configured to have different parallel processing queues n, with $2 \leqslant n \leqslant 64$.
- According to SIP protocol, the UA has to REGISTER at a SIP server before it can INVITE others entities or receive INVITE messages. Therefore, REGISTER is the first essential step for the whole process and our experiments are focus on this step. We have configured our UA to send continuously REGISTER messages from our local network to external SIP proxies. The external SIP register addressed are given to the UA in textual representation, as such our proxy has to resolve the domain before it can forward any request.
- The attacking tool is configured to send crafted messages containing hard resolvable domain names to the local outgoing SIP proxy. It is configurable by the attacking interval i seconds between two attacking messages.

To measure the proxy performance, we send out 5000 register messages from our UA and count the number of responses (*r*) our local proxy can process. If we can get any kind of response from a remote SIP server, it means the domain name of the server has successfully been resolved by our proxy. 50 INVITE messages to different external SIP servers which are randomly chosen will be send in 1 second and every outgoing message from our UA is routed thought our local SIP proxy, Without any attack, we have measured *r* to be close to 5000, while under attack is considerably lower.

Table 1 shows the variables of experiment.

Table 1: Experiment variables

Variable description	Variable symbol
Parallel processing queues of the proxy under attack.	*n*
Time interval between two attacking messages sent from the attacking tool to our local proxy.	*i*
Number of reply messages received by our UA	*r*

We repeated all experiments are based on this test bed 10 times and calculated the mean values.

5. LIMITED ATTACK MITIGATION POSSIBILITIES

DoS attacks are the main suspects for causing lost availability. Traditionally, the general rule of alleviating impact from a DoS attack is to trace the source of the attack and block the traffic from it as close to the source as possible. However, it is difficult to apply this method on SIP network since the SIP protocol runs at the application layer [1], thus, back tracing will be very costly in this case. We explore here different possibilities that might allow server operation even in case of an attack.

5.1. Reduced FQDN Usage

The source of the described attack is the usage of fully qualified domain name addresses in SIP messages. Hence, FQDN should not be used unless necessary. The standard provides means to reduce FQDN usage in VIA header that indicate the path taken by the request so far. That is, each proxy that receives a request adds its URI to the list. The receiver of the request adds the VIA list to its replies and then sends the reply to the topmost VIA entry. Each proxy receiving the reply removes the VIA entry indicating its URI and forwards the reply to the new topmost entry. To avoid the need for resolving a URI included in a VIA entry of a reply, it is possible to add a "received" parameter to the VIA entry of the request with the numeric IP address of the sending entity, thus eliminating the need to resolve this URI after receiving a reply. However, the effective of this mechanism to defense against users with dedicated malicious intend is quite limited. Although we can use numeric IP address instead of FQDN in the VIA header, we cannot avert using domain name of extent SIP server in the REGISTER and INVITE head. It will be inconvenient for user to remember the IP address of external server. Therefore, the attackers still can launch the attack by filling unresolvable domain names in the REGISTER or INVITE head.

5.2. Scalable Server Design

Another option is to design the receiving proxy in a scalable way to increase performance. Traditionally, two concepts are to be considered: *Synchronous scaling through parallel processing* and *asynchronous scaling through message processing interruption*.

5.2.1. Synchronous Scaling through Parallel Processing

To reduce blocking effects, SIP proxies, including SER, use parallel message processing. Such a SIP proxy is extended to operate with threads or parallel processes with each process or thread responsible for processing one message synchronously. Such a design is depicted in *Figure 5*. Here a core part only acts as a message scheduler distributing incoming messages between the processes. Each process is then responsible for parsing the message, initiating any DNS requests or requesting the execution of an application and finally forwarding the message. State information can be shared among the processes using some form of shared memory.

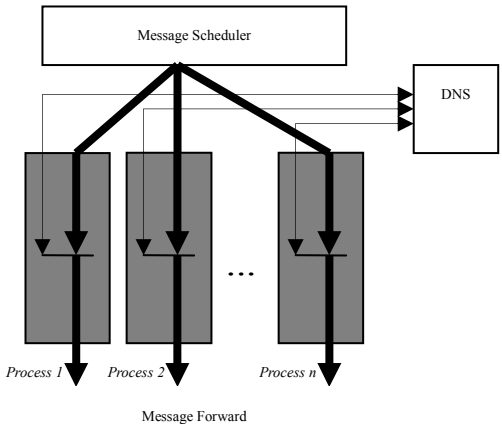


Figure 5: parallel process design of the SIP proxy

This kind of DoS prevention is commonly known as

over-provisioning. Resources, which can easily be exploited by an attacker, are extended, thus lowering resource exploitation possibilities. This generally leads to a race condition, where the attacker increases the attack rate to cope with more powerful servers. With distributed DoS attacks, this can be easily achieved [23].

To evaluate the performance of parallel processing under the attack, we perform an experiment based on our test bed with different parallel processing queues *n* and different attack intervals *i*.

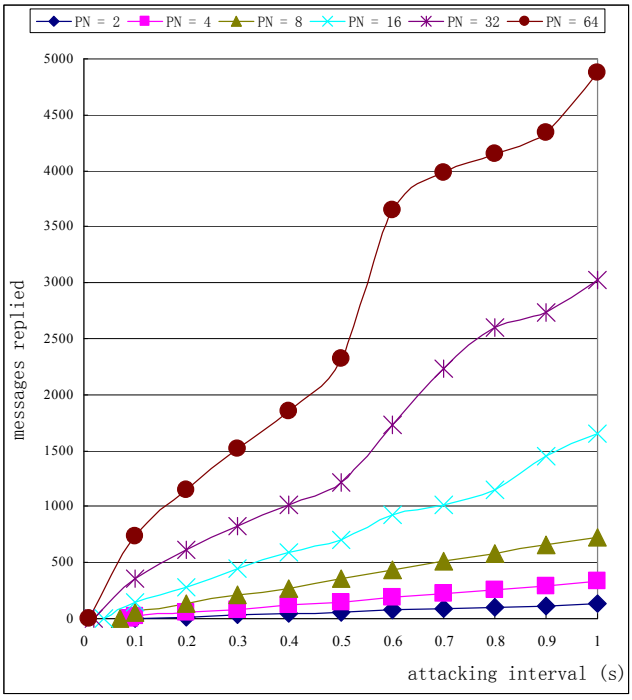


Figure 6: the performance of SER with different processes and attacking interval under attack

The result is shown in *Figure 6*. With few parallel processing queues ($n \leqslant 8$) less than 20% of all potential messages can be processed, even with only one attack message per second. 64 processing queues are needed to adequately cope with the same attack speed of one malicious message per second. However, decreasing the attacking interval down to 0.001 seconds (1000 attack message per second), even 64 parallel processing queues are completely starved.

Generating 1000 messages per second is easily achieved with a DDoS attack, where an attacker controls hundreds of slave machines [3]. For example, Hussain et al. demonstrated an attack scenario with 100.000 malicious messages per second [23]. In this scenario, under this attack, even a proxy with 64 or more processing queues would be totally blocked. On the other side, more parallel processes cost more memory and CPU resources, possibly leading to system overload and hence another type of DoS. In this experiment, we draw a conclusion that it is quite limited improved to mitigate the attack by simply forking more processes.

5.2.2. Asynchronous Scaling through Message Processing Interruption

Another option is to design all requests to external servers as non-blocking. That is, after issuing a DNS request the server would not wait until an answer for the request was received but would queue the request in an event queue, save the data of the transaction, set the current operation on hold and move to processing the next request. When a reply for the request arrives the main process is notified and the broken transaction is scheduled to continue, thus eliminating a DNS blocking scenario. The procedure is shown as *Figure 7*. However, since the states of unfinished domain name resolving requests have been saved, the implementation complexity and memory requirements increase considerably. The server must support effective state suspend and resume capabilities, as each new DNS requests requires to completely storing the actual state into memory, and returning this state upon DNS resolve notification. We have shown that a SIP attack launched at a SIP proxy running on a machine with 8GB of RAM all memory can be depleted in about 30 seconds [24].

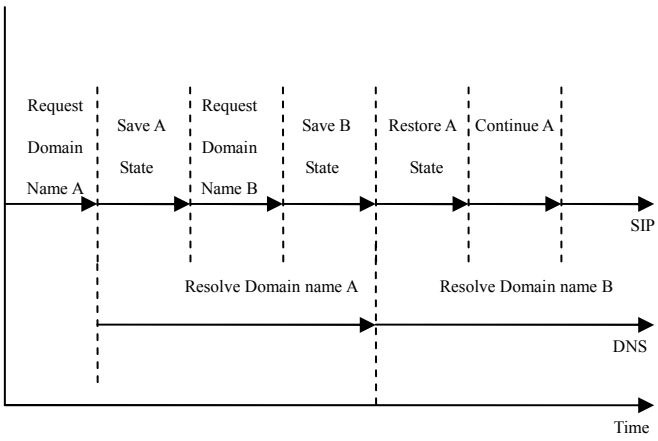


Figure 7: the procedure of asynchronous scaling design

6. NON-BLOCKING CACHE DESIGN

Our experiment with parallel message processing has shown some benefits in message processing, although such a design does not prevent DoS attacks on the DNS system. Additionally, we've seen that under attack it is not feasibly to try to resolve all possible malicious domain names when an actual attack is underway.

Hence, it is mandatory that the proxy under attack conditions does not attempt to resolve every domain name. This raises two questions: How can such an attack be detected? And how would be a countermeasure to other, non-malicious users?

6.1. Attack Detection and Prevention

The goal of a DNS attack is to force a proxy as long as possible to wait in the operating system's domain resolve call (e.g. *gethosbyname*). With this in mind let us assume a SIP proxy *S* with *n* parallel processes as described in section 5.2.1. We define:

$$S_q(t) = \begin{cases} 1, & \text{a domain resolve call in process queue } q \text{ but not returned at time } t, \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

We also define *H* as an indicator how many processes are concurrently resolving a domain name in time *t*, with

$$H = \sum_{q=1}^{n} S_q(t), \quad (2)$$

Hence the proxy will absolutely be blocked when $H = n$. To guarantee non blocking proxy operation, the following relation has to be met: $H < n$ at any time t. To achieve this we define a minimum operation threshold *m*, where *m* is reasonably small and $m < n$. Whenever $H \geqslant R$, where $R = n - m$, the proxy is informed that further DNS resolve request will have a high possibility to cause a DoS due to proxy blocking. As a consequence, the proxy will not try to resolve any domain names whenever $H \geqslant R$. Instead, the proxy assumes this address to be unresolvable, and continues its operation. As soon as $H < R$, the proxy will again perform DNS lookups.

As an example take a proxy with n=16 processes. We leave m=1 "emergency process". Whenever R=15 or more processes are blocked due to DNS lookup, the remaining 1 process will not perform such lookup, until at least 1 process is concurrently free for further operation.

6.2. Operational Consequences

This design has some consequences on non-malicious, regular users of this proxy. As long as $H < R$, proxy behaviour is not affected, with the proxy serving both an attacker and all regular users. In the other case however, no request from a regular user will be served. A self-inflicted DoS is thus created, with a similar effect as intended by the attacker.

To remedy this situation, we introduce a dedicated DNS cache for the SIP proxy. A DNS cache answers to DNS resolve requests from the SIP proxy. It saves the results of the previous DNS queries, if the SIP proxy tries to resolve the same address a second time, the stored result in the cache can be returned instead initiating another time consuming query, as such also speeding up general system performance.

While different operating systems already provide DNS caches, they lack dedicated features for optimal usage in a SIP network. As described, a SIP entity uses additionally DNS records to locate other proxies, including NAPTR / SRV records, while a general operation system DNS cache does not consider such records for caching. Furthermore, a dedicated SIP DNS cache needs a specialized replacement policy, as it should clean out some records and cache some new records.

Combining the non-blocking design with a dedicated SIP DNS cache will effectively counter DNS attacks while keeping negative side effects on regular users to a minimum:

- As long as $H < R$ there should be no visible effect on regular users.
- In case of an ongoing attack, many regular users won't be affected: Current connections will be kept, REGISTER updates are executes without delay. Also, often new requests could still be served as long as the destination address is available in the cache.
- Only requests to destinations not currently in the cache will be dropped. These requests can not be handled at the moment.

As a result, this solution allows reduced operability under attack conditions. The amount of negative side effects on regular users mainly depends on the implementation of the caching replacement policy.

6.3. Operational Performance

To test the feasibility of such a design, we have implemented a prototype which operates with SER. The DNS cache prototype is to be implemented for the following three considerations: (1) the implementation of "emergency process", which will only look up DNS record internally instead of forwarding requests to external DNS servers whenever $H \geqslant R$; (2) The prototype should cache both regular DNS entries (DNS `A` records) and DNS `SRV` records; (3) Apply different cache replacement policies such as first in first out (FIFO), least recently used (LRU), least frequently used (LFU) and Time Cost to replace old records[25] [26], which we will examine further on.

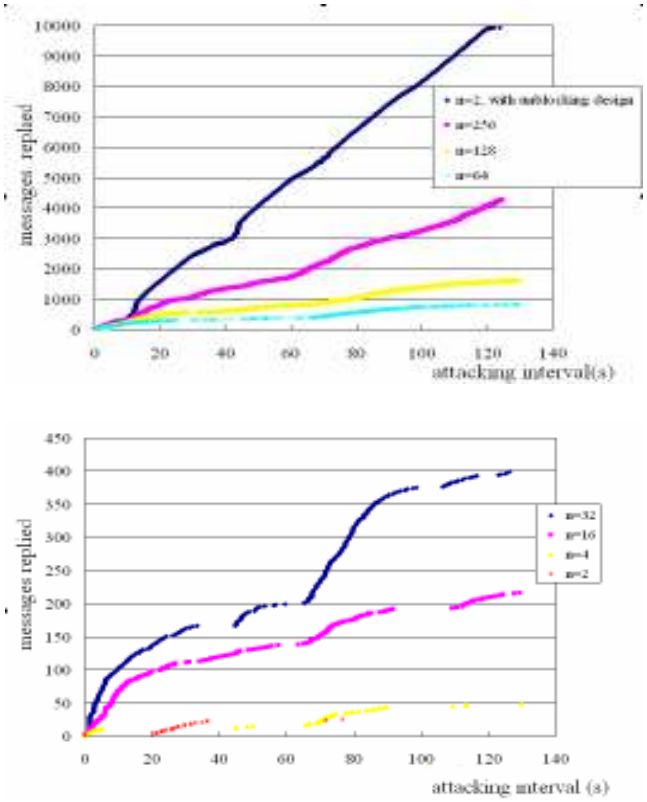


Figure 8: unblocking evaluation of the cache prototype. With it, almost no replied message is lost even as n=2. Without it, most messages are lost whatever n is.

6.3.1. Unblocking Test

In order to verify the unblocking ability of our prototype, we've run several endurance tests with our described attack script, generating 10,000 messages containing difficult resolvable domain names with 10 ms delay between each message. As an example see *Figure 8* which shows the number of resolved messages over time at the SIP proxy (n=2, 4, 16, 32, 64, 128, 256) without DNS cache and proxy (n=2) with cache (optimal). Every test case runs about 140 seconds.

Looking at *Figure 8*, we can see that overall performance depends on the number of active processing queues, with n=256 showing the best results with about 4300 of the 10.000 messages processed. Contrary to this the proxy can process under 50 requests with n=2. Hence, even with 256 processes, the performance of the proxy is still very limited in case of an attack. After deploying our cache solution (FIFO policy, 80 cache entries) the proxy answers nearly all 10,000 requests, and we found the result is independent of the number of activated processing queues n.

Furthermore, we can also see from the picture that without the cache deployed, the proxy will stop working for several seconds during the testrun, e.g. at 90s < t < 110s with n=16. During this 20 s the proxy is blocked completely, as all 16 queues are waiting for a response from the DNS server. Comparing this with the figure from the testrun with the cache, we can witness the continuous operation of the proxy.

6.3.2. Cache Replacement Policies Evaluation

As the he number of cache entries (e) can not practically cope with the unlimited number of possible domain names, we have to find a way to optimally use the limited number of cache entries. Even a large cache entry storage can easily be allocated by an attacker with randomly generated invalid domain names, thus not leaving space for usable records. Hence, this might cause another DoS due to the misconfigured DNS cache. Additionally, keeping DNS entries for a long time or even indefinitely in the cache might be exploited by an attacker to launch a DNS cache poisoning attack [27]. We design a cache replacement policy to counter these threats, where the cache learns about new entries and replaces old ones.

We consider four established policies; *First-in, First-out* (FIFO), *Least Recently Used* (LRU), *Least Frequently Used* (LFU) are well-known cache replacement strategies for paging and web scenarios [26] [20]. Considering that the time cost of looking up different domain names maybe different, we consider also a *Time Cost* (TC) strategy (see *Table 2*).

Table 2: different caching replacement policies

Name	Primary Key
FIFO	Entry Time of Object in Cache
LRU	Time Since Last Access
LFU	Frequency of Access
TC	Request Time Cost

Generally, all replacement strategies applied on a queue of cache entries. The newest record is inserted into the head of the cache queue, while entries are deleted from the tail of the cache queue when the size of the cache storage is exceeded.

- For the FIFO policy, except for newest and oldest entities, all records will be moved towards the tail by one when a new record enters the queue.
- LRU policy is similar with FIFO, with the difference that whenever one record within the cache is accessed, it will be moved directly to the head of the queue.
- With LFU policy the DNS records are arranged by the

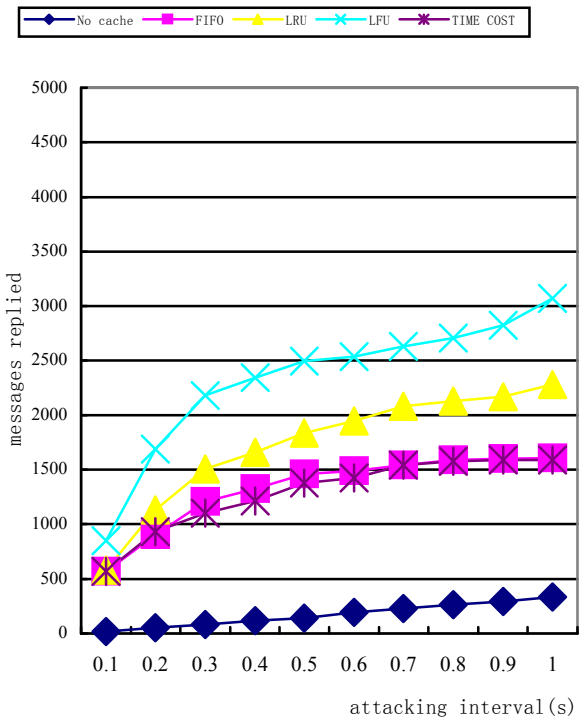
frequency of their usage of DNS records. The higher the frequency, the closer entries are located toward the head of the queue.

- TC policy is similar to LFU, but the queue is ordered by the time cost of the DNS lookup time of an entry. The goal is to keep entry with a higher lookup time available in the cache. The higher the lookup time cost of en entry, the closer it is to the head of the queue.

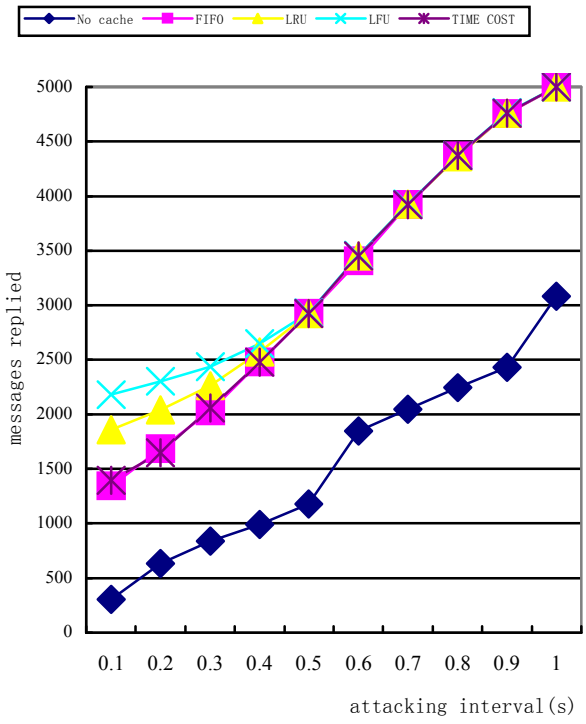
We repeated the experiment in 5.2.1 with these four caching strategies twice, one time with n=4 parallel processing queues at the proxy and then again with n=32. With a testrun we have found out that within our experiment we will need e=270 entries in our DNS cache to hold all domain names of the 100 contacted SIP proxies. The reason for this higher number is that sometimes root nameservers have to be contacted first before the actual proxy name can be resolved. (e.g. *ns2.mydyndns.org* will be contacted automatically before looking up *iptel.org*). Cache replacement strategies can only be tested if the number of possible entries is lower than the total number looked up in the experiment. To compare the effectiveness of the replacement strategies we arbitrarily set the entry number of our cache (e) to 80 which is reasonably lower than the possible 270 entries. The result can be seen in Figure 9.

Part (a) of Figure 9 shows the performance of the different caching strategies for n=4 in case of an attack. We can see clearly that DNS caching with any caching strategy yields better performance than without DNS caching. The improvements vary on the attack interval and the used replacement policy, with Least Frequently Used (LFU) algorithm giving best results. The figure shows from 17% successful responses out of 5000 (at i=0.1 s) up to 61% successful responses (i=1 s). In comparison, figures for the uncached experiment are from 0.4% (i=0.1 s) up to 6% (i=1 s).

In part (b), showing n=32, we can see a different result. Especially from attacking interval i>0.5 s, the performance of all four algorithms are generally equal. This seems to be due to the increased processing power of the proxy with n=32, as also in the uncached experiment we see a clear increase in successful resolve requests. In this case, the attack is simply not powerful enough to block the proxy. However, with increased attack speed again LFU shows best results. We measure 44% successful responses (i=0.1 s) up to nearly 100% successful responses (i=1 s), in comparison to 6% (i=0.1 s) / 60% (i=1 s) for the experiment without cache.



(a) The proxy is with 4 parallel processes.



(b) The proxy is with 32 parallel processes.

Figure 9: the performance of SIP proxies equip with different cache replacement policies under attack

Furthermore, we further decreased the attacking interval to 0.02s which totally blocks the proxy without cache (0%

successful responses); the proxy still manages to process 1936 (38%) messages with the assistant of our LFU cache.

We estimate that higher performance of the LFU algorithm is caused by the difference of DNS data structure of external server and internal cache. As we mentioned above, the external DNS database is organized as a tree structure [10] while our DNS cache is organized as a two-dimensional hash table, and if the system try to find the DNS record of a SIP provider (e.g. *iptel.org*) from external DNS server, the root name server (e.g. *ns2.mydyndns.org*) of it has to be found at first. After that, both the record of *iptel.org* and *ns2.mydyndns.org* will be cached. The root name sever in the cache is important because it may be helpful to find other SIP providers later, but it is still not as important as the SIP providers record, which is we really care about. Next time, when another DNS request of *iptel.org* arrives, the system will look up the *iptel.org* record in the cache directly without looking up its root name server because it is no need to find the root in a two-dimensional table. Therefore, for the same inquiry, the record in cache of *ns2.mydyndns.org* will not be used as long as the record of *iptel.org* is still in cache. As a result, the proxy will work better if we keep as much as the record of sip providers which is used frequently and remove the record of root servers which is used seldom. Fortunately, the behavior of LFU policy is to keep the most frequently used records in the cache and wash out the least frequently used once so that it is easy to withhold the recorder of SIP providers which is we really wanted and kick that of the root nameservers out. The other three policies are not based on this factor. Therefore, we consider that LFU replacement policy is the most qualified policy than other three for the SIP infrastructure.

6.3.3. Evaluate the Number of Entries of Cache

In the last experiment, we survey the relationship of caching performance in relation to caching entries *e*. As we mentioned before, the number of cache entries is limited while the amount of the domain names in the real world is almost unlimited so that the cache is impossible to accommodate all the domain names in the world. On the other hand, more cache entries will cost more memory and CPU resources to support them. To investigate that how the number of cache entries will affect the performance of caches, we set n=4 and x=0.5 (s), and evaluate with different number of cache entry based on the test bed. The result is shown in *Figure 10*.

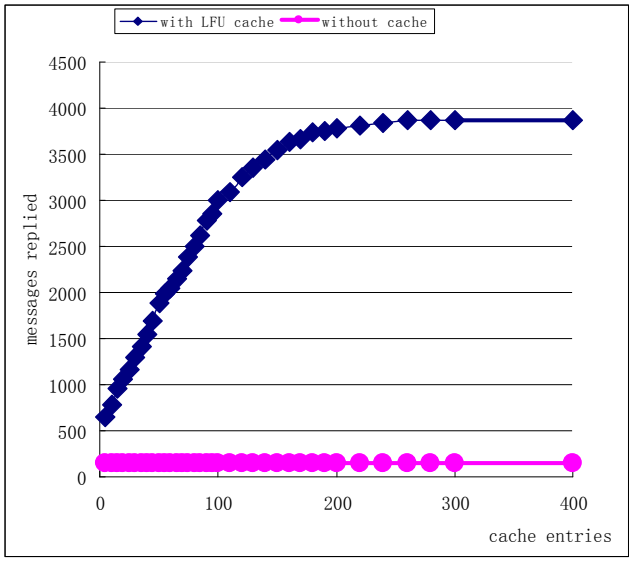


Figure 10: the performance of proxy with different number of cache entries under attack

From the *Figure 10*, we can find that when even there are only 5 entries in the cache (least situation in the experiment), the proxy still works better than without cache. But the more cache entries it is, the better performance the proxy does. When the number of cache entries is less than 150, the number of replied messages grows sharply with the increasing of cache entries while the growth becomes suddenly relaxed when the number of cache entries is among 150 and 270. Finally, the curve totally ceases to ascend when the cache owns more than 270 entries or so, which is quite similar as we anticipated because there are 270 different domain names involve in the test, and there is no cache record will be replaced if the cache entries are enough ($e \geqslant 270$). Furthermore, we also set cache entries number to 500, 600 until 2000 and all the results are around 3870.

7. CONCLUSION AND FUTURE WORK

Denial of Service attacks can limit SIP server operation to a large amount. In this paper we have evaluated attacks on SIP servers that utilize unresolvable DNS names in SIP messages. Such attacks might be interesting for malicious users, as such SIP messages can easily be crafted and already a low amount of such message can reduce server operation by far. With chance, the attacker can block one processing queue of the server by one single message for five seconds.

Even over-provisioning the servers with massive parallel operation and asynchronous DNS lookup capabilities can not successfully counter such attacks. We have shown that in this case server operation will not be blocked; however, an even more

severe Denial of Service was easily achieved through message flooding eventually leading to Out-of-Memory errors and following server shut-down.

Hence, we postulate that a server will never be able to be on par with a DoS attack. Instead, intelligent design is necessary to alarm the server on an imminent attack under way with an "emergency operation mode" in case of attack.

We've demonstrated such an alternate operation mode, which tries to take into account two imminent contradicting conditions, that a server should stay operational even under attack load that normally would consume all server's resources, and still serve all regular users without adverse effects or them. Our solution based on non-blocking name resolving together with DNS name caching, keeps the server operational while still providing substantial service to regular users.

This experiment has been conducted with a limited number of SIP proxies; hence the figures can not accurately reflect a real-life scenario where requests need to be processed with a higher number of domain names. In further work we'd like to investigate virtualization techniques to simulate testruns that closer match the real life.

8. ACKNOLEDGEMENT

This work has been partly conducted in the European Union funded IST-COOP-5892 project SNOCER (www.snocer.org).

9. REFERENCE

[1] F. Cao and S. Malik, "Security Analysis and Solutions for Deploying IP Telephony in the Critical Infrastructure", *Security and Privacy for Emerging Areas in Communication Networks, 2005*, Workshop of the 1st International Conference on.

[2] M. Handley and A. Greenhalgh, "Steps Towards a DoS-resisteant Internet Architecture", *SIGCOMM'04 Workshops, Sep 3, 2004, Portland, Oregon, USA*.

[3] J. Mirkovic, S. Dietrich, D. Dittrich and P. Reiher, "*Internet Denial of Service: Attack and Defence Mechanisms*", 2004 - Prentice Hall PTR Upper Saddle River, NJ, USA.

[4] L. Gordon et al., "*CSI/FBI Computer Crime and Security Survey*", Computer Security Inst., 2004.

[5] D. Sisalem, J. Kuthan and S. Ehlert, "Denial of Service Attacks Targeting a SIP VoIP Infrastructure: Attack Scenarios and Prevention Mechanisms", *IEEE Network Vol. 20, No. 5 - Special Issue on Securing VoIP*, Sep. 2006.

[6] D. R. Kuhn, T. J. Walsh and S. Fries, "*Security Considerations for Voice over IP Systems"*, Recommendations of the National Institute of Standards and Technology, January 2005.

[7] D. Geneiatakis, G. Kambourakis, T. Dagiuklas, C. Lambrinoudakis and S. Gritzalis, "A Framework for Detecting Malformed Messages in SIP Networks", *Local and Metropolitan Area Networks, 200, LANMAN 2005,* the 14th IEEE Workshop on.

[8] E. Y. Chen, "Detecting DoS Attacks on SIP System", *VoIP Management and Security, 2006, 1st IEEE Workshop on,* 3 April 2006.

[9] H. Sengar, D. Wijesekera, H. Wang and S. Jajodia, "Fast Detection of Denial of Service Attacks on IP Telephone", *Proceedings of IEEE IWQoS'2006, New Haven, CT*, June 2006.

[10] M. Nassar, R. State, O. Festor, "Intrusion Detection Mechanisms for VoIP Applications", *3rd Annual VoIP Security Workshop*, Jun 2006, Berlin, Germany.

[11] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, R. Spark, M. Handley, E. Schooler, "RFC 3261: SIP -- Session Initiation Protocol", 2002.

[12] H. Schulzrinne, J. Rosenberg, "The Session Initation Protocol: Internet-centric signaling", *Communications Magazine, IEEE*, Oct 2000.

[13] P. V. Mockapetris, "RFC 1034: Domain names – concepts and facilities," Nov. 1987.

[14] P. V. Mockapetris, "RFC 1035: Domain names – implementation and specification," Nov. 1987.

[15] J. Rosenberg, H. Schulzrinne, "RFC 3063: SIP -- Locating SIP Servers", June, 2002.

[16] J. Peterson, H. Liu, J. Yu and B. Campbell, "RFC 3824: Using E.164 Numbers with the Session Initiation Protocol (SIP)", 2004.

[17] A. Gulbrandsen, P. Vixie and L. Esibov, "RFC 2782 - A DNS RR for specifying the location of services (DNS SRV)", Feb 2000.

[18] J. Jung, E. Sit, H. Balakrishnan and R. Morris, "DNS Performance and the Effectiveness of Caching", *IEEE/ACM Transactions on Networking (TON)*, Jan 9, 2002.

[19] V. Pappas, Z. Xu, S. Lu, D. Massey, A. Terzis and L. Zhang, "Impact of Configuration Errors on DNS Robustness", *SIGCOMM'04 Workshops,* Sep 3, 2004, Portland, Oregon, USA.

[20] Berkeley Internet Name Domain (BIND), Open source domain name source, http://www.isc.org/index, accessed at Oct, 2006**.**

[21] SIP Express Router, Open source SIP proxy, http://www.iptel.org/ser,

[22] SIPp, traffic generator, http://sipp.sourceforge.net/, accessed at 21st Aug 2006.

[23] A. Hussain, J. Heidemann and C. Papadopoulos, "A Framework for Classifying Denial of Service Attacks", *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003.

[24] D. Sisalem, S. Ehlert, et al. "General Reliability and Security Framework for VoIP Infrastructures" *Technical Report SNOCER-D2.2*, Sep 2005, www.snocer.org.

[25] A. Silberschatz and P. B. Galvin, *Operating Systems Concepts, fourth ed. Reading,* Addison-Wesley, 1994.

[26] C. Aggarwal, J. Wolf, and P. Yu, "Caching on the World Wide Web", *IEEE Transactions on knowledge and data engineering, Vol. 11, No. 1,* Jan 1999.

[27] T. Olzak, "DNS Cache Poisoning: Definition and prevention", http://www.infosecwriters.com/text_resources/pdf/DNS_TOlzak.pdf, accessed at Oct, 2006.

Conclusion

First of all, I would like to answer the three research questions mentioned in the proposal.

- How to find a proper method to mitigate the effect of DoS attack via DNS request?

During the thesis work, I developed the proposed prototype, compared the performance of it with other methods and evaluated in the simulated SIP environment. The result of it is obviously better than other methods from different aspects. Therefore, up to now, the special DNS cache is the best method to solve this problem in our research scope.

- Which factors of DNS cache and SIP proxy (e.g. caching replacement policy, cache entry number, parallel processes number of proxy, etc) are useful to deal with this problem?

During the thesis work, I measured six different parameters: the number of emergency processes, hash table entry number, caching replacement policy, cache entry number, parallel processes number of proxy and attacking interval. These six factors are the most general in SIP architecture despite of hardware and platform. Through these experiments, I found last four factors quite affect the performance of SIP proxy. The relationship of these four factors and the performance of SIP proxy are as follow, (Assume there are not enough cache entries to accommodate all records): (1) LFU policy could help cache to work better than other replacement policies. (2) If there are enough CPU and memory resource, the more parallel processes of SIP proxy, the better performance of SIP proxy. (3) If there are enough CPU and memory resource, the more cache entries, the better performance of SIP proxy. (4) The longer attacking interval, the better performance of SIP proxy.

- Which kind of combination of the useful factors is the most efficient?

Mentioned in the last question, there are four factors affect the performance of SIP proxy. The most efficient is supposed to be a SIP proxy with more parallel processes, a DNS cache with LFU cache replacement policy and more cache entries. Whereas the number of parallel processes and cache entries should be controlled according to the server's capacity.

In summary, Denial of Service attacks can limit SIP server operation to a large amount. In this paper we have evaluated attacks on SIP servers that utilize unresolvable DNS names in SIP messages. Such attacks might be interesting for malicious users, as such SIP messages can easily be crafted and already a low amount of such message can reduce server operation by far.

Our tests also show that over-provisioning the servers with massive parallel operation and asynchronous DNS lookup capabilities as well as reducing the usage of DNS names in the SIP messages can not successfully counter such attacks. That is while such measures can improve the performance under attack a severe attack will also manage to block the server at some point. By combining DNS caching with a blocking threshold after which no new DNS requests are issues, we have shown that a SIP server can continue working even under a heavy attack. This threshold can be either the percentage of blocked process in a SIP server designed to process messages in a synchronous manner. For servers processing messages in an asynchronous manner this threshold could represent the percentage of used memory. Finally, in case the used cache can not accommodate all possible DNS names, our experiments suggest that using a least frequently used replacement strategy for the cache has resulted in the highest hit rate.

Future Work

In this research, we have verified that the cache mechanism could mitigate this kind of attack by a certain amount. But it is not a perfect solution: there are still lots of sessions lost when the attacker decrease the attacking interval. And on the other side, with the new subsystem (DNS cache) introduced, new security issue comes to life, such as DNS cache poisoning, etc. Furthermore, the test bed we based on is quite limited so that we cannot simulate all the SIP session traffic, (e.g., there is only REGISTER message sent to proxy, not INVITE, ACK and BYE messages). In the future work, we will start following three aspects:

- Enhancement the defence system. For example, in the research, we already know that caching replacement policy, cache entry number, parallel processes number of proxy and attacking interval affect the performance of cache and proxy. We can determine the former three factors from the server side but the attacking interval depends on the attacker, not us. Fortunately, it is possible to decrease attacking speed by some intelligence filter technique. Now we devote to combine this solution with other IDS and IPS in the SIP infrastructure and constructing a scalable defence system.
- Consider the new threat introduced by DNS subsystem. For example, we will exam the possibility of cache poisoning in our prototype and try to mitigate the threat by improving TTL (Time to Live) algorithm of DNS records. We will develop a DNS cache poisoning tool which could maliciously modify cache records and compare with the previous result. We also will take cache TTL as a important parameter in the experiments.
- Accurate the research result. In this thesis, the test bed only covering SIP REGISTER traffic environment, which is only the first step of SIP processing. Therefore we plan to construct the new test bed to simulate the whole process of SIP session building, include REGISTER, INVITE, ACK, BYE, etc. We tried this before, while the configuration complexity increases considerably because every SIP provider has their own restrict rules on outgoing request. To perform such a test, it is quite difficult to rely on the SIP providers in the real world. We plan to construct a new test bed based on virtualized machines. We will still use SIPp UA to simulate traffic, consisting of REGISTER and INVITE requests with successive OK responses. We will use one SIPp UA as the caller and another as the callee. After caller, callee registered in the server, caller would send INVITE request to callee continuously. The architecture showed in Figure App.1.

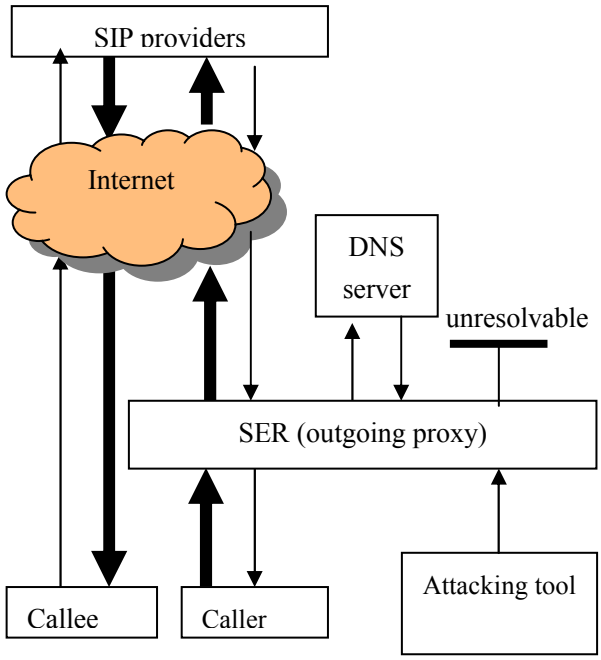


Figure App.1: an architecture of new test bed

Appendix A ----*The Design of DNS Cache for SIP Proxy*

The DNS cache prototype is to be implemented for the following three considerations: (1) keep at least one SIP proxy child process unblocked; (2) add SRV, NAPTR record into the cache; (3) apply different cache replacement policies (FIFO, LFU, LRU, Time Cost)to knock out old records.

1. View as a whole

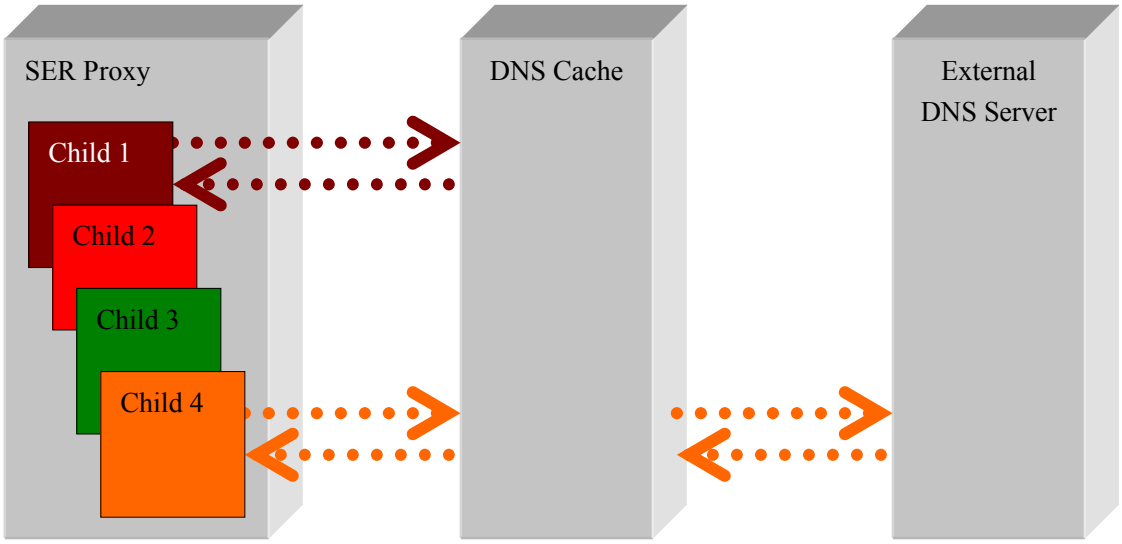


Figure App.2: the fundamental role of DNS cache in the SIP environment

Three parts are involved in the system, SER proxy, DNS cache and external DNS server. SER proxy is taken as a server which helps users to build a connection with another remote SER proxy. Generally speaking, it is forked into several parallel processes to deal with the request of multiple users in the same time. Furthermore, sometimes local SER proxy searches another remote SER proxy via domain name. If the local SER proxy receives a request to send SIP messages to Alice@ABC.com, it will try to find the target IP address of the SIP server by domain name ABC.com at first. Of course, the SER proxy can execute a DNS request to upper DNS server directly, but it is very awful efficiency and easily suffers from Denial of Service attack since the DNS server maybe take several seconds to reply. DNS cache is introduced to solve the problem. It will save some DNS records temporally in the cache, and if the SER proxy has to make a DNS inquiry, it will search the DNS cache at first, then external DNS server if there is no result in the cache.

In the Figure App.1, the Child 1 performs a DNS request to DNS cache at first, and gets the record in the cache. While the child 4 cannot find its record in DNS cache, as a result the cache requests it from upper DNS server. The record will saved in the cache and sent back to SER proxy after the upper DNS server replies.

2. Architecture Design of DNS Cache

There are three parts in the DNS cache, (1) communication interface; (2) packets dispatch; (3) cache structure. Communication interface is used to exchange packets with the children processes and the upper DNS server. For DNS, UDP and port 53 should be adopted. Two sockets will be created when the DNS cache initialized, one to communicate with SER, and another with External DNS server.

To avoid complex structure and bugs, single-process will be adopted instead of multiple-process. But on the other side, the children process of SER proxy shouldn't be blocked when they send request in the same time. Fortunately, packets dispatch will be introduced to solve this problem. When DNS cache receives a request packet form a child process, it will save the IP address and the port of the packet source in a record and assign a unique ID to the record. Next, the DNS

cache will replace the packet ID with the record unique ID and transmit the packet to upper DNS Server. Then, the DNS cache will not wait for the reply of upper DNS server and can freely deal with requests from other children. Until upper DNS server replies, the DNS cache will find the target address and port by the packet ID from the structure and transmit the reply packet back. Furthermore, if there is no response received from external DNS server to a certain request for up to 5 seconds, the state will be removed and send back unresolvable.

ID	TTL (S)	Source Address
…	….	…
3456	2	{10.1.20.9, 2531}
3457	3	{10.1.20.45, 4638}
3458	3	{10.1.20.9, 3056}
3459	4	{10.1.20.7, 8930}
…	….	…

Table App.1: an example of the request dispatch table in the cache

To solve the first problem we mentioned at the beginning: "(1) keep at least one SER proxy child process unblocked.", the DNS cache will know how many parallel processes the SER proxy has and save it in a global counter. Every time the DNS cache receives a request from SER proxy, the counter will minus one. By contrast, the counter will add one if the cache gets a reply from upper DNS server or a request is timeout. When the counter is less than two, which means only one unblocked child process left, the DNS cache will not transmit the request packet to upper DNS server, instead, it will only look up locally and reply immediately.

Since there are a lot of SRV and NAPTR application in SIP environment besides A, we should save all these three kind of records into the cache; the node structure (the unit of cache) is designed as follow,

```
struct node {
    struct node *next; //used for cache algorithm
    struct node *prev; //used for cache algorithm
    struct node *hash_next;  //used for hash table
    unsigned short type;  //used to the type of record, A,SRV, NAPTR
    time_t ttd;          // time to die
    union {
        struct all_addr addr;    //A record
        struct srv_record srv;   //SRV record
        struct naptr_record naptr; //NAPTR record
    }
    Unsigned int weight; //used for cache algorithm
    unsigned short flags;    //reserved
    char name[ MAX_NAME];    //the domain name
};
```

The nodes exist as two kinds of data structures in the same time, (1) hash table; (2) double linked list.

Hash table is used to accelerate the search speed. The nodes which have the same digest value will share the same hash entry. Hash entry should be found before look up the node to locate the entry. When a new node is added into the hash table, the hash entry will be calculated and it will be inserted as the first node of the entry.

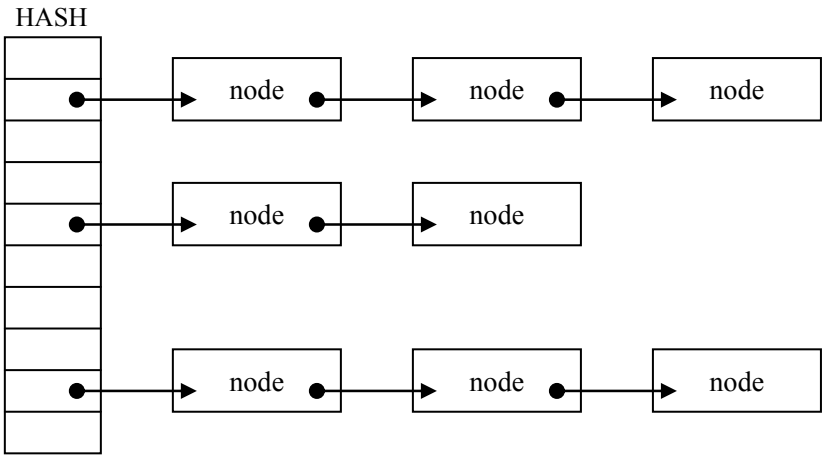


Figure App.3: an overview of the cache entries organized as hash table

When the cache receives a reply of new DNS request, the new record will be inserted. When the number of nodes exceeds the limit, some nodes will be knocked out via cache algorithm. To achieve it, a queue of double linked list will be adopted.

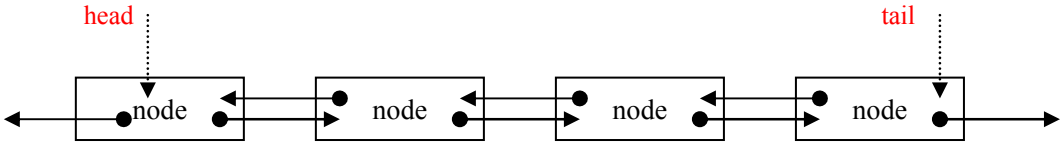


Figure App.4: an overview of the cache entries organized as double linked list

- FIFO: the new node will be inserted after head and the old one will be knocked out from the tail.
- LFU: the queue is sorted by the visit frequency, the head points to the most frequent one and the tail points to the least frequent one. If the frequency is the same, the least recently used nodes will be knocked out.
- LRU: every new node or just visited will be moved to the head. The node pointed by tail will be knocked out.
- Lookup Time Consuming: the queue will be sorted by the lookup time consuming, and the least time consume node will be knocked out.

Appendix B ----*the scenario script of SIPp*

SIPp is a free Open Source test tool / traffic generator for the SIP protocol. In our test bed, it was used to simulate the normal REGISTER traffic. I wrote the following script to create the REGISTER scenario.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">
<scenario name="RIGISTER">
    <send>
        <![CDATA[
            REGISTER sip:[field1] SIP/2.0
            Via: SIP/2.0/UDP [local_ip]:[local_port];branch=[branch]
            Cseq: 3349 REGISTER
            To: "nick" <sip:[field2]@[field1]>
            Expires: 1800
            From: "nick" <sip:[field2]@[field1]>
            CALL-ID: [call_id]
            Content-Length: 0
            User-Agent: SIPp
            Event: registration
            Allow-Events: presence
            Contact: "nick" <sip:[field2]@[local_ip]:5067;transport=udp;>;methods="INVITE"
        ]]>
   </send>
    <recv response="478" option="true">
    </recv>
    <recv response="408" option="true">
    </recv>
</scenario>
```

The code between <send> </send> indicates sending a REGISTER message to a randomly selected external SIP provider via SER proxy. Next, the SIPp will perform a DNS inquiry to locate the IP address of the external SIP provider. In our experiment, there are four possible situations in the following:

- Due to attacking, all the parallel processes of SER are blocked. Therefore, no process could handle this REGISTER message so that the message will be ignored by SER proxy. As a result, nothing will be replied to SIPp.
- Due to attacking, only one emergency processes is not blocked, and the REGISTER message will be handled by it. No result is found in the internal cache and “478 cannot resolve the domain name” will be replied to SIPp.
- The domain name of the external SIP provider in the REGISTER is successfully resolved and the message is forwarded to the provider. But Due to attacking, the proxy may not receive the reply from the provider. In this case, “408 timeout” will be sent to SIPp.
- The domain name of the external SIP provider in the REGISTER is successfully resolved and the message is forwarded to the provider. After while, the proxy get the reply from external provider and forward the reply message to SIPp.

  In this scenario, if SIPp get a response from external providers, a successful session will be counted. Since “478” and “408” is the reply generated by proxy, not from external providers, therefore we have to filter them (see the two <receive> </receive> parts in the script).

Appendix C ----*A difficult-resolvable DNS flooding tools*

First of all, let us recall that whenever a user requests a domain resolve, there are generally two cases to distinguish:

- *The DNS server knows the name mapping*. The name server might know the mapping because it is the authoritative name server for this domain. As such, all mappings for the domain are preconfigured for this domain server. The server might also know the name because it has resolved this address previously. Generally, in this case the mapping is still stored in the server's internal cache.
- *The DNS server does not know the name mapping*. In this case the server will issue a recursive request to other name servers that might be able to provide an answer. The server will eventually receive a response, either containing the valid mapping or an error message that no mapping is possible. In the former case, the mapping will be stored in the server's internal cache for a limited period of time. The names server can also set a time limit for the query. If no answer is received within this limit, the address is considered unresolvable.

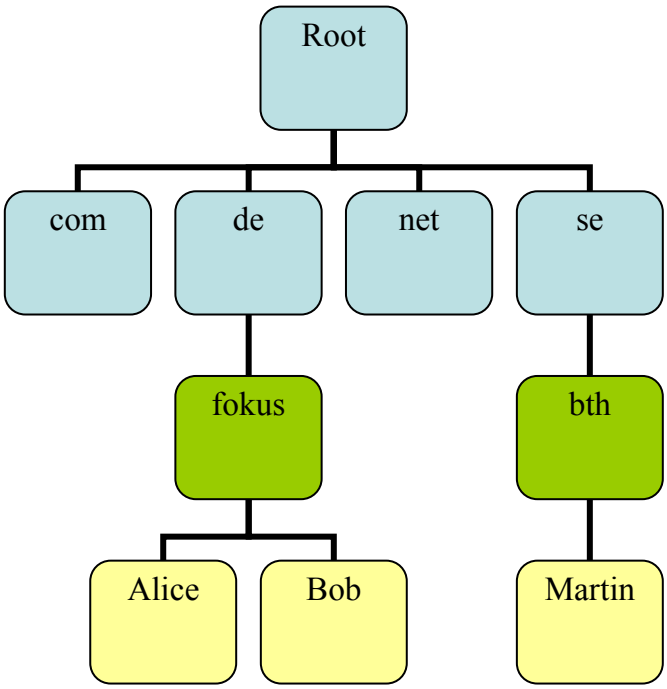


Figure App.4: an example of the organization of DNS

For example, showed in figure App.4, suppose if there are two local DNS server, fokus and bth, Alice is a host of fokus and Martin is a host of bth. When Alice would like to look up the IP address of Martin, Alice will send request to fokus at first. If no matched record returns, focus will reply Alice:"I don't know, please ask other servers". The DNS request will be forward to other DNS servers recursively, for example, de, se, bth. Finally, bth will answer the IP address of Martin.

Now let us imagine two possible situations as follow:

- If the access bandwidth from bth to fokus is low, much time has to be spent on connection.
- If the DNS server of bth is mis-configured or there are plenty of hosts in bth domain, the processing of looking up will cost much time.

An attacker could easily exploit this vulnerability by fill random host name in front of bth domain automatically, such as (sdf.bth.se, asfgsf.bth.se). Therefore, a difficult-resolvable DNS flooding tool can be created.

Appendix D --- *Experiment Setup and validation*

1. Experiment 1

   In this test, I took parallel processing queues *n* and different attacking intervals *i* as variables. The experiment verified that it was quite easy to launch such a DNS flooding attack to reduce the availability of service. It was also showed that the solution of increasing parallel process number of the proxy was limited. I did these experiments with *n* processes of proxy and *i* attacking interval for ten times. The Figure App.5 shows the average calculated from them.

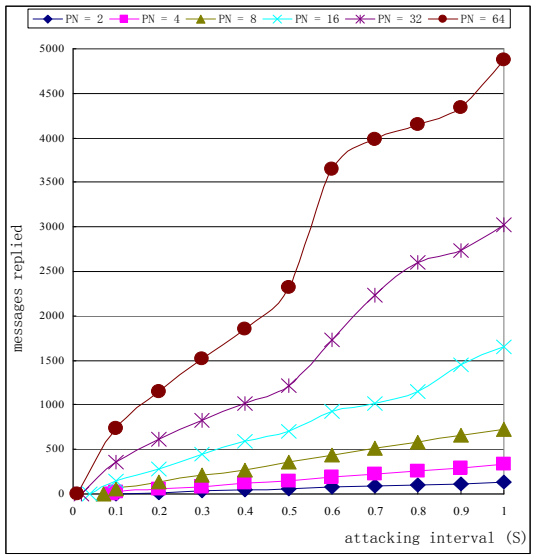


Figure App.5: processing performance of the local proxy for different processing queues n and varying attacking intervals i

2. Experiment 2

   Experiment 2 could verify the unblocking ability of our prototype. This experiment is the only one not based on the test bed we mentioned before. I wrote a script on the server so that it could log timestamp in a log file when the server answered a SIP request. I did these experiments with *n* processes of proxy and *i* attacking interval in two situations, with cache solution and without cache solution. Without cache, the result is the same despite of different process numbers. Before the experiment, I did a pre-test and found the result is irrelative with emergency number.

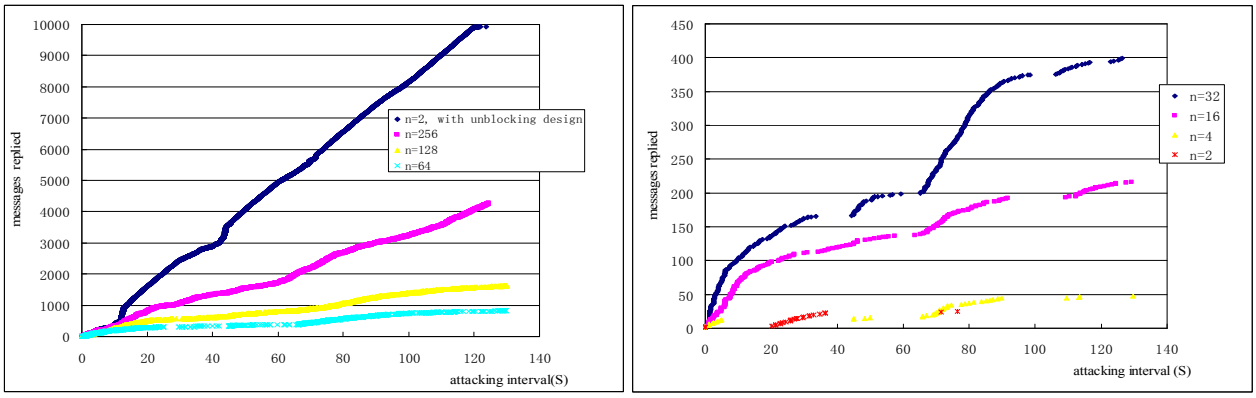


Figure App.6: Message processing capabilities. The more parallel queues n are configured, the more messages can be processed during an attack. With the DNS cache implementation nearly all messages can be processed independently of n.

3. Experiment 3

The purpose of experiment 3 is to find out which cache replacement policy is more qualified in this situation. The parameters include cache replacement policy, attacking interval, process number of proxy. The experiment is based on the test bed. Before the experiment, I also did a pre-test and found 270 cache entries will be occupied to resolve these 100 providers, therefore, the maximum cache entries number was intentionally set to 80 so that the replacement policy worked. The picture shows the results of average values of ten times repeated test.

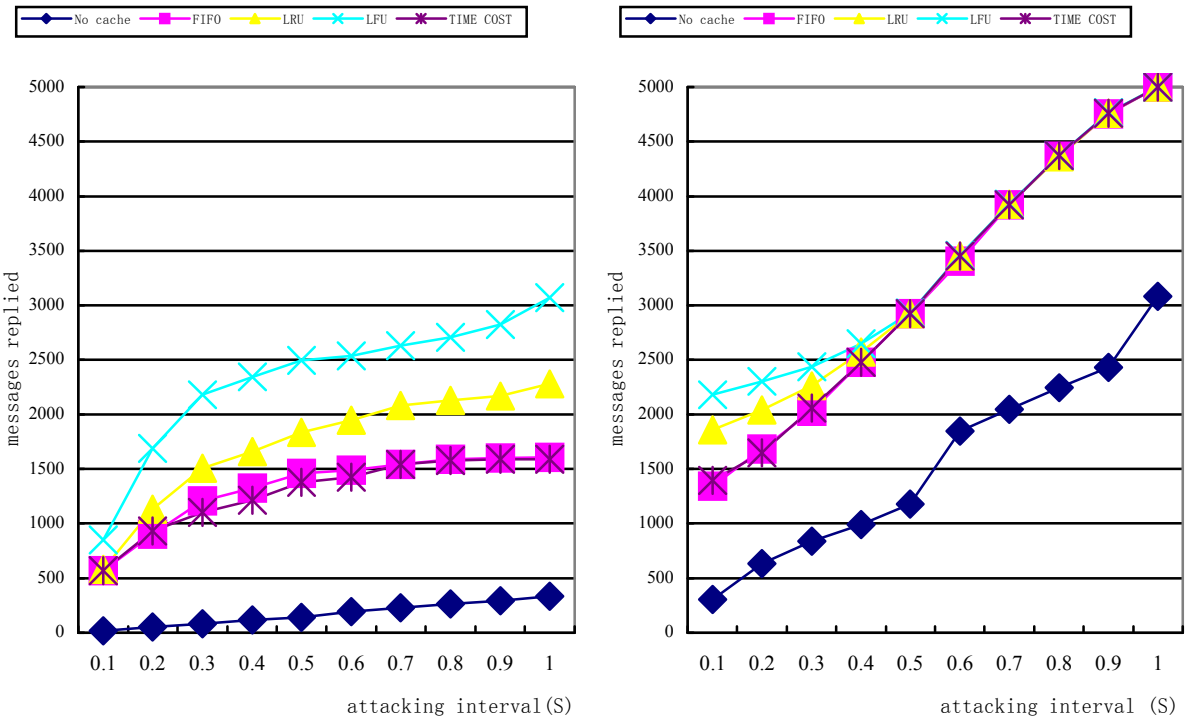


Figure App.7: performance of SIP proxies equipped with different cache replacement policies under attack

4. Experiment 4

The last experiment is to investigate the relationship of cache entry number and the performance of the proxy. The cache entry number is the only variable. The picture shows the results of average values of ten times repeated test.

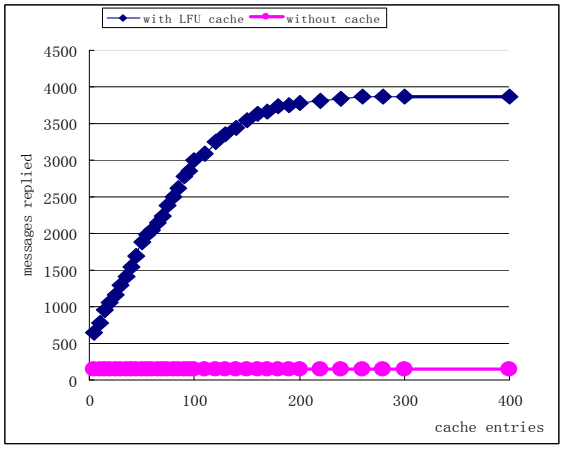


Figure App.8: performance of proxy with different number of cache entries under attack

Appendix E ----*100 external SIP providers used in the experiments*

1 iptel.org
2 voiz.irepublics.com
3 calamar0.nikotel.com
4 bluesip.net
5 fwd.pulver.com
6 sipgate.co.uk
7 sipnumber.net
8 proxy01.sipphone.com
9 voiptalk.org
10 sip.voipuser.org
11 register.voipgate.com
12 sip1.sippal.com
13 sip.inphonex.com
14 babble.net
15 sip.freeipcall.com
16 proxy1.sipsnip.com
17 proxy2.sipsnip.com
18 xchange.terracall.com
19 sip.webphone.com
20 sip.voipfone.co.uk
21 sipbroker.com
22 sip.myvoipaccount.net
23 sip2go.com
24 nadiz.com
25 sip.broadvoice.com
26 voip.eutelia.it
27 sip.stanaphone.com
28 voztele.com
29 sip.libretel.com
30 sip.alteline.com
31 voxalot.com
32 callcentric.com
33 sip2.bbpglobal.com
34 sip.net2phone.com
35 sip.sinapsys.net
36 sip.televoip.no
37 voipuser.org
38 sip.callunion.com
39 sip.gradwell.net
40 sip.telic.net
41 proxy.lax.broadvoice.com

proxy.dca.broadvoice.com
proxy.mia.broadvoice.com
proxy.atl.broadvoice.com
proxy.chi.broadvoice.com
proxy.bos.broadvoice.com
proxy.nyc.broadvoice.com
calamar.nikotel.com
sip.ucs.sfu.ca
gw1.voicepulse.com
gw2.voicepulse.com
register.zivvaoffice.com
sip.starshipcorp.com
thekompany.com
sip.3c-hungary.hu
sip.3c-russia.ru
sip.inphonex.com
sip.callclarity.net
bbtele.se
sip.winradius.net
sip.mobitus.com
sip.webphone.com
sip.voipbuster.com
northamerica.sipphone.com
sipdr.quantumvoice-sip.com
sip.force9.net
draytel.org
sipdr.quantumvoice-sip.com
Voip-co2.teliax.com
sip.unlimitel.ca
sip.peoplecall.com
voip.cascotec.com
talk.rabbitpoint.net
sip.ixcall.net
atlas-east.vonage.net
sip.varphonex.com
sip.televoip.no
sip.voise.com.au
sip.internetphoneco.com
sip.1und1.de
sip.simply-connect.de
sipgate.de
sip.gmx.net
iphone.freenet.de
proxy.de.sipgate.net

86 sip.tiscali.de
87 sip-gmx.net
88 sip.sipservice.eu
89 voipgateway.org
90 sipdiscount.com
91 tel.t-online.de
92 freephonie.net
93 sip.schlund.de
94 callcentric.com
95 at43.tuwien.ac.at
96 sip.backbone.ch
97 ch03.sip-fon.eu
98 proxy.digisip.net
99 deu1.purtel.com
100 sip.a1.net