Spoof Detection for Preventing DoS Attacks against DNS Servers

Dr.T.Pandikumar[1], Yehenew Mekonen[2]

[1]Ph.D. Department of Computer & IT, College of Engineering, Defence University, Ethiopia
[2]M.Tech. Department of Computer & IT, College of Engineering, Defence University, Ethiopia

---***---

Abstract - *The Domain Name System (DNS) is a hierarchical decentralized naming system for computers, services, or any resource connected to the Internet or a private network and it is a critical element of the Internet infrastructure because of this it needs a good security mechanism. Domain Name System (DNS) Service is the basic support of Internet, which security plays a vital role in the entire Internet. Even a small part of the DNS infrastructure being unavailable for a very short period of time could potentially upset the entire Internet and is thus totally unacceptable. The original motivation for this seminar title is most solutions are model based on intrusion detection. Unfortunately, because DNS queries and responses are mostly UDP-based, it is vulnerable to spoofing-based denial of service (DoS) attacks, which are difficult to defeat without incurring significant collateral damage. The key to prevent this type of DoS attacks is spoof detection, which enables selective discarding of spoofed. DNS requests without make vulnerable the quality of service to legitimate requests. On this seminar title we have going to see a comprehensive study on spoof detection strategies for protecting DNS servers from DoS attacks. These strategies all create some form of cookies for a DNS server to check if each incoming request is indeed from where the request packet says it is from, but vary in performance overhead, transparency and deployment complexity. Which implemented all of them as a firewall module called DNS guard. Measurements on the current DNS guard prototype show that it can deliver up to 80K requests/sec to legitimate users in the presence of DoS attacks at the rate of 250K requests/sec.*

Keywords: DNS spoof detection, Defense against spoofing-based DNS, DoS attacks, ANS, DDoS

1. INTRODUCTION

The Domain Name System (DNS) is a critical component of the Internet infrastructure, because most network services and applications require a translation step from domain name to IP address to just send the packets out. As a result, even a small part of the DNS infrastructure being unavailable for a short period of time could have a significant rippling effect on the rest of the Internet. However, common DNS queries and responses use UDP as their transport protocol. The combination of the simplicity of the DNS protocol and its use of UDP makes DNS extremely vulnerable to spoofing-based Denial of Service (DoS) attack. Unlike TCP, UDP does not use three-way handshake procedure to start a connection and therefore has no way to be sure that a UDP packet indeed comes from where the packet's source address indicates. Worse yet, a DNS server only sees one UDP query and replies with one UDP response for most DNS interactions. Therefore it is not possible for a DNS server to ascertain the identity of the requesting host at the DNS level, either.

Denial-of-service attack is a type of attack on a network that is designed to bring the network to its knees by flooding it with useless traffic. Many DoS attacks, such as the Ping of Death and Teardrop attacks, exploit limitations in the TCP/IP protocols. For all known DoS attacks, there are software fixes that system administrators can install to limit the damage caused by the attacks. But, like viruses, new DoS attacks are constantly being dreamed up by hackers [6]. There are two possible DoS attack strategies against DNS servers. The first is to send a large number of requests to a DNS server to overload it.

There are several spoof detection strategies and implemented all of them in a firewall module called DNS guard, DNS Guard is a family of DNS-based security services that protects your network and your users from harm. Because it operates within the network, DNS Guard protects users without requiring installation of any software and protects all types of IP-enabled devices, including desktops, tablets and smartphones [7]. Among the three most popular Internet protocols used by network applications, TCP, UDP and ICMP, only UDP-spoofing attacks do not have an adequate countermeasure yet. TCP SYN flooding was addressed by SYN cookie. ICMP spoofing can be contained by limiting its rate. Among UDP-based network services, only DNS is truly indispensable and needs to be open to the whole public. Therefore, it is worthwhile to develop spoof detection techniques specifically tailored to DNS traffic.

1.2 Statement of the problem

While the methods describe appear to be effective at detecting spoofed packets, they are not perfect. The complexities of the modern computer networks can create situations that complicated detecting spoofed packets. Also, an attacker who knows that a system is being monitored for spoofed packets may craft more sophisticated packets to defeat the spoofed packet detector.

A common problem existing in this preventive mechanism is that the reconstruction of attack path becomes quite complex and expensive when there are a large number of attackers (i.e. for highly distributed DoS attacks). Also, these types of solutions are designed to take corrective action after an attack has happened and cannot be used to stop an ongoing DDoS attack. In addition this seminar focus only preventing DOS attack on DNS server on the other hand there is so many attacks which is not covered on this seminar.

1.3 Objective of the Research paper

The general objective of this research is to detect spoofed packet to prevent DOS attack against DNS server by creating some form of cookies for a DNS server to check if each incoming request is indeed from where the request packet says it is from, but vary in performance overhead, transparency and deployment complexity. Which implemented all of them as a firewall module called DNS guard.

1.4 Significance of these research works

The main significance of this research work is to prevent the DNS server from DOS attack :

I. creating some form of cookies for a DNS server to check if each incoming request is indeed from valid users
II. Implemented all of them in a firewall module called DNS guard, which is designed to be deployed without modifying protected DNS servers or DNS requesters.

The main significance of this research work is to show using some mechanism how to protect DNS server from denial of service attack.

2 DESIGN AND IMPLEMENTATION

2.1 Overview

The DNS infrastructure comprises three types of components: the *stub resolver*, *local recursive server* (LRS), and *authoritative name server* (ANS). The stub resolver is typically implemented as a library on an end-user machine. It is not sophisticated enough to do everything that a local recursive server can. Whenever a network application requests a name resolution, the stub resolver simply sends a recursive DNS request to the local recursive server. The local recursive server (LRS) is usually set up for an organization, e.g., a department in a university. LRS provides two main functionalities. First, it is capable of serving recursive requests. To answer a recursive request, LRS sends one or multiple iterative requests (message 3, 5, and 7) to multiple ANSs. Second, LRS can cache the answers from ANSs, and queries ANSs only when it cannot answer with its cache.

The authoritative name servers (ANSs) maintain a name-address mapping database. ANSs are shown: root, *com*, and *foo.com*. For example, to resolve the address of *www.foo.com*, one of the root DNS servers is queried. Currently there are thirteen well-known IP addresses for root DNS servers world-wide. The root DNS server returns the name and IP address of the *com* domain's name server using an NS (Name Server) record and A (Address) record in message 4. Then the LRS queries the *com* domain name server for the IP address of the *foo.com* domain name server, which in turn answers the IP address of the host *www.foo.com* using an A (Address) record. In the above process, the root server and the *com* domain name server only provide referral information (NS records and records for the ANSs of the next level of domain). The *foo.com* name server provides the final authoritative answer.

The general strategy to ascertain the source of a DNS request is to send a cookie to the requesting host after receiving the first request, and require the requesting client to attach the cookie to all subsequent requests. There are two design issues: (1) How to return a cookie to an LRS? (2) How to trick an LRS into embedding a cookie in every request. We explore three schemes in this section. The first scheme is to embed cookies into legitimate DNS messages, where the cookie could be represented by a *referral's name* or a *part of an IP address*. The second scheme is TCP-based DNS, where the cookie is represented by TCP's sequence number. The third scheme is to modify DNS by explicitly introducing a cookie exchange procedure. The first two schemes do not require modifications to LRS, whereas the third scheme does.

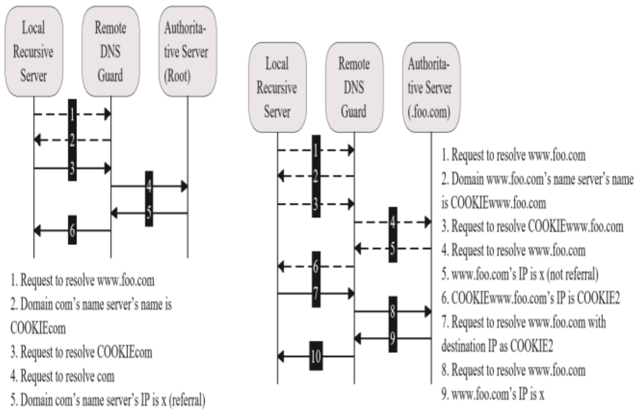

Local Recursive Server
Remote DNS Guard
Authoritative Server (Root)
1
2
3
4
5
6
1. Request to resolve www.foo.com
2. Domain com's name server's name is COOKIEcom
3. Request to resolve COOKIEcom
4. Request to resolve com
5. Domain com's name server's IP is x (referral)
Local Recursive Server
Remote DNS Guard
Authoritative Server (.foo.com)
1
2
3
4
5
6
7
8
9
10
1. Request to resolve www.foo.com
2. Domain www.foo.com's name server's name is COOKIEwww.foo.com
3. Request to resolve COOKIEwww.foo.com
4. Request to resolve www.foo.com
5. www.foo.com's IP is x (not referral)
6. COOKIEwww.foo.com's IP is COOKIE2
7. Request to resolve www.foo.com with destination IP as COOKIE2
8. Request to resolve www.foo.com
9. www.foo.com's IP is x

Figure 1: Using NS name and IP to embed cookie in traditional DNS
(A) Embedding the cookie in the NS name for Referral answers (B) Embedding the cookie in fabricated NS name and IP for non-referral answers.

2.2 DNS-based: Embedding Cookies in DNS Messages

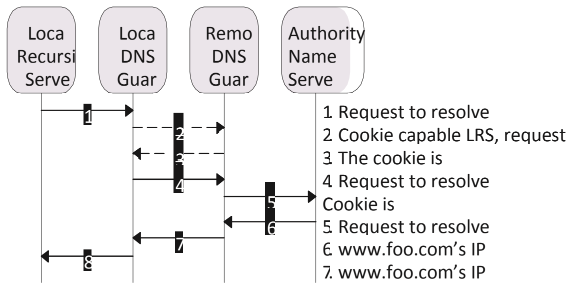
The ANS can return two kinds of answers: a referral answer or a non-referral answer. A referral answer provides information about the ANSs in the next level of the domain name hierarchy. A non-referral answer is any answer that is not a referral.

1) Referral Answer: Embedding Cookie in NS Name: The referral information in DNS is represented in two types of resource records. The first type is the NS (Name Server) record, which provides the name of an ANS. The second type is the A (Address) record, which provides the IP addresses of an ANS. If an LRS only receives the name of an ANS, it issues another query to find out the ANS's IP address and query the ANS. The key idea here is to exploit the fact that an LRS is capable of executing further queries when the LRS only receives the name of an ANS. Basically this algorithm replaces the real name of an ANS with a fabricated name in which the cookie is embedded. That is, an LRS never sees the real NS records. Instead, a fabricated NS record is received for each domain.

There are several issues in the above design. First, instead of two packets and one round trip time (RTT), there are four packets and two RTTs for each interaction. Fortunately, this overhead can be minimized by setting a large Time To Live (TTL) value for the fabricated NS record (i.e., *COOKIEcom* in the above example). This doesn't break the TTL design of the original referral information because the TTL of the ANS's IP address does not change. In normal operations, the fabricated NS records rarely expire. So the LRS can query an ANS with a cookie embedded NS name directly when the ANS's IP address expires. That is, an LRS's cache could eliminate message 1 and 2 during most operations. Second, in each domain, there are usually multiple ANSs each with a distinct name and IP address. In the above scheme, the LRS never sees the real names of the ANS but fabricated names with cookies embedded. Fortunately, one name can be mapped to multiple IP addresses. By returning multiple IP addresses for a fabricated domain name, the LRS can still use multiple ANSs. Third, the current scheme assumes that the ANS will return the IP address of the next-level ANSs in message 5.But this is not always true. Sometimes an ANS only returns the name of the next-level ANSs. Fortunately, standard DNS delegation practice requires each next-level domain to provide both the name and IP address of its ANS. By adding the IP address to the ANS's zone file, all ANSs can return the IP address of the next-level ANSs in message 5. Fourth, the cookie is encoded in the form of *COOKIEcom* in the above example. This means both the cookie and the original question are encoded in one *label* whose length is limited to 64 bytes according to RFC 1035. Fortunately, legitimate domain names are much shorter than 64 bytes. As a result, there is plenty of room to embed cookies.

2) Non-Referral Answer: Embedding Cookie in NS Name and IP: The above scheme does not work if the ANS returns non-referral information, e.g., an A (Address) resource record for the queried name.Instead, we introduce a second cookie to achieve 1 RTT for the best case. The key idea is to fabricate an ANS for each non-referral answer. For each fabricated ANS, two records are faked: an NS record and an A record. Each record embeds one cookie.

3) Summary: In summary, this DNS-based scheme embeds cookies in fabricated NS records for referral answers. For non-referral answers, a fabricated ANS (an NS record and an A record) is created for each



non-referral request. The scheme can be implemented as a firewall module and is totally transparent to both ANS and LRS. Neither ANS nor LRS needs to be modified. This transparency is the key advantage of this scheme. However, it pays its price by creating more state and/or latency overhead. The solution for referral answers doesn't have extra state or latency overhead. The cookie embedded NS records need to be cached by LRS anyway. The maximum latency is 2 RTT and only happens when an LRS contacts an ANS for the first time. The solution for non-referral answers is less good. Encoding cookies as the IP address of a fabricated ANS limits the security strength to the size of the subnet where the DNS guard is deployed. The maximum latency is 3 RTT when the LRS contacts an ANS for the first time. Moreover, each name requires a fabricated ANS (an NS record and an A record) being cached in the LRS, which means multiple cookies are stored duplicate when there are multiple names cached from the same ANS.

2.3 TCP-based: Transparently Fall Back to TCP

The key idea of this scheme is to exploit the DNS truncation notification mechanism to notify a requesting LRS to use TCP as the transport protocol. Because TCP uses three-way handshake to establish a connection, it is difficult to spoof the source IP address for DNS requests. Normally DNS uses UDP and limits the UDP message size to be 512 bytes or less. For message size larger than 512 bytes, ANS replies with a truncation flag. Then the LRS will automatically initiate a TCP connection and send the request again on the TCP connection.

2.4 Modified DNS: Extending DNS Protocol

The idea of this scheme is to extend DNS protocol to explicitly support cookies so that best efficiency can be achieved. Meanwhile, it serves as the optimal baseline to benchmark the other two schemes. The extension is backward compatible with traditional DNS. It uses a similar method as DNSSEC (DNS Security, RFC 2535) to extend the DNS protocol.

(a) Extending DNS protocol to support cookies in the request.

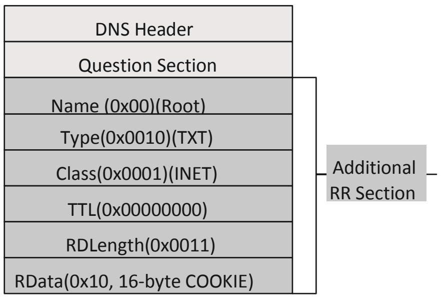


Figure 2: The method to extend DNS protocol to incorporate cookies in the request.

As in Figure 3, the DNS protocol is extended and both ends need to be modified to understand the new protocol. To facilitate deployment, we developed two firewall modules which can be deployed independently of the DNS software being used. The local DNS guard is deployed in front of LRS and the remote DNS guard is deployed in front of an ANS. The LRS is not changed so it sends out traditional DNS requests (message 1). The local DNS guard intercepts the request and checks if it knows the cookie to the destination ANS. If it has cached the cookie, message 2 is skipped and message 4 is sent directly. Otherwise, message 2 is sent to request a cookie from the ANS. The remote DNS guard intercepts message 2 and returns message 3 with a cookie that is a function of the source IP address of the DNS request. The local DNS guard then sends message 4 with the cookie. The remote DNS guard checks the validity of the cookie. Only request with valid cookie is passed to the ANS. In message 5, the cookie information is removed, so the ANS doesn't see any cookie extension. Message 6, 7 and 8 are forwarded without any change.

2.5 Cookie Design

The cookie is computed as follows. For each DNS request whose source IP address is source_ip, its cookie is: c = MD5(source_ip, key). Each DNS guard holds a 76-byte secret key. No key distribution is needed because only the DNS guard needs to know it. The 76-byte key is concatenated with the 4-byte source IP address and the resulting 80-byte plain text is fed to the MD5 hash function whose minimum input is 80 bytes. The MD5 function generates a 16-byte hash value as the cookie c for source_ip. For an attacker to attack an ANS, he needs to know the correct cookie c for each spoofed source_ip. This requires the attacker to know the ANS's key, whose large size makes it difficult to be compromised.

The COOKIE size is a tradeoff between the security strength and traffic amplification. In Figure 2, message 1 may be a spoofed request. Message 2 will reply with referral name (NS record). The NS record causes message 2 longer than message 1 thus traffic amplification. The NS record is similar to the TXT record shown in Figure 3(b). The total increase is 24 bytes. Because the minimum size of a DNS request is around 50 bytes (IP packet size), the traffic amplification effect is at most 50% for DNS-based scheme. For the modified DNS scheme, the whole cookie c is stored directly in the message. Since the request and the reply have the same size, there is no traffic amplification.

2.6 The Big Picture

Three spoof detection schemes are presented in this section. They all require some form of initial handshaking to ascertain the IP address of the requesting client and generate a cookie that serves as a credential in subsequent interactions. The first scheme (DNS-based) embeds cookies into existing DNS protocol messages. The second scheme (TCP-based) uses TCP's sequence numbers as cookies. The third scheme (modified DNS) introduces a cookie exchange process explicitly and requires extending the DNS protocol.

The TCP-based scheme uses TCP to transport DNS requests and responses. The main disadvantages of this approach are long latency and large processing overhead. Neither the DNS-based scheme nor the TCP-based scheme requires modification to LRSs. The modified DNS scheme extends the DNS protocol and thus needs to extend LRSs. But it is secure and efficient in cookie storage, and incurs small request latency without complicating protocols or amplifying traffic.

2.7 Attack Analysis

The goal of spoof detection is to protect an ANS from being bombarded by a spoofing-based DoS attack or becoming a traffic amplifier. One prevents traffic amplification by designing short response packets when a DNS request's source IP address is not verified yet. In the DNS-based scheme, the traffic amplification ratio is less than 50%. For the TCP-based and modified DNS scheme, the truncation response and cookie response are of the same size as the DNS request, so there is no traffic amplification at all. Moreover, Rate-Limiter1 could control the DNS response rate to top requesters, and thus makes it difficult to use ANS as a traffic amplifier.

Another attack is to guess the value of a cookie. The first way is to brute-force all possible values of a victim host's cookie. The cookie range for NS name can be easily larger than 4 billion, and the cookie range for the modified DNS scheme is 16 bytes. The fabricated NS and IP variant of the DNS-based scheme has the smallest cookie range. For small networks, the range of R_y may be less than 254, and therefore it is not difficult to enumerate all possible values of y. One attack strategy is to send an attack request to the ANS with a guessed y value. While the attack traffic is going on, the attacker does a normal DNS query to the ANS to probe its performance and see if the guessed value is correct. For this attack to succeed, the attack rate must be sufficiently high to saturate the ANS. Fortunately,

Yet another attack is to brute-force the key used in generating the cookie. We use the MD5 hash function with a 76-byte key. Up to now, it is generally believed that it is very difficult to obtain the input to an MD5 hash function from its output. Explicitly enumerating all possible keys also seems impractical.

One can also obtain a host's cookie with respect to an ANS by sending a DNS request to the ANS and sniffing the network for the corresponding response. However, this requires the attacker to be on the same subnet as the spoofed host. Even when an attacker successfully obtains a host's cookie, not much damage can be done because Rate-Limiter2 can throttle an individual host's request rate. This is the same case as when the attacker mounts a DoS attack using public or zombie computers.

3. PERFORMANCE EVALUATION

	DNS-based		**TCP-based**	**Modified DNS**
	NS Name	Fabricated NS Name/IP		
Cache Miss	21.0	32.1	34.5	22.4
Cache Hit	11.1	11.3	33.7	10.8

3.1 Testbed Setup

We set up a testbed to evaluate the performance of the three DNS spoof detection schemes. The testbed consists of six nodes: one remote DNS guard, one local DNS guard, one ANS and three LRSs. The ANS and the three LRSs are connected via the DNS guards, which handle cookies in DNS requests and deliver only valid ones to the ANS. The local DNS guard is only used when testing the modified DNS scheme. The DNS guards are DELL 600SC with 2.4 GHz CPU, 512MB memory, and a dual-port Intel gigabit Ethernet card with 33-MHz 64-bit PCI interface. The ANS and 3 LRSs are DELL 400SC machines with 2.26 GHz CPU, 512MB memory, and an Intel gigabit Ethernet card with 33-MHz 32-bit PCI interface. All machines run Linux 2.4.31. The DNS guards work in the router mode and the spoof detection mechanisms are implemented in the iptable module. The ANS and LRSs run BIND (version 9.3.1) or a DNS simulator program to test the throughput of the DNS guard. Unless specified otherwise, each reported number is an average of 20 measurements. The average round-trip time (RTT) between the LRSs and the ANS on the testbed is 0.4 msec.

3.2 DNS Request Latency

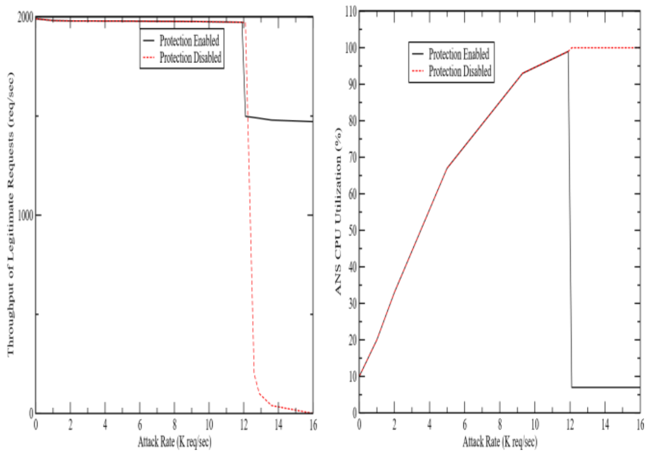
In this test, the ANS is on a university campus network and the LRS is on a cable-modem network that is connected to ANS through the Internet, and the average round-trip time (RTT) between the ANS and the LRS is 10.9 msec.

Table I shows the average DNS request latency under different spoof protection schemes for the first access (cache miss) and subsequent accesses (cache hit). When an LRS interacts with the DNS guard for the first time, it needs to get a cookie, which takes multiple RTTs. After the first access, the LRS can cache the cookie and reuse it in subsequent accesses, which need only one RTT to complete.

Therefore, the latency of most DNS requests is mainly limited to RTT. By capturing packets at the ANS, the DNS guards and the requesting LRS, we find that the combined processing time inside the DNS guards, the ANS and the requesting LRS is less than 1 msec even for cases that need 3 RTTs. Among the spoof detection schemes studied, the TCP-based scheme incurs the worst latency, and the DNS-based schemes are comparable to the modified DNS scheme.

Table 1 : Average DNS request latency (msec) for different spoof detection schemes.

The average RTT between the requesting LRS and the ANS is 10.9 msec.



(a) The throughput of legitimate requests.

Figure 4: Throughput and CPU utilization of an ANS running BIND 9 when the DNS guard is turned on and off.

The DNS guard can protect the legitimate request throughput of a BIND server and reduce its CPU utilization when it is being attacked.

3.3 BIND Throughput under Attack

In this subsection, we report the DNS request throughput of a BIND-based ANS with and without the DNS guard when it is being attacked. All machines run BIND version 9.3.1. First we measure the maximum throughput of BIND, which is 14K requests/sec when UDP is used, and 2.2K requests/sec when TCP is used.

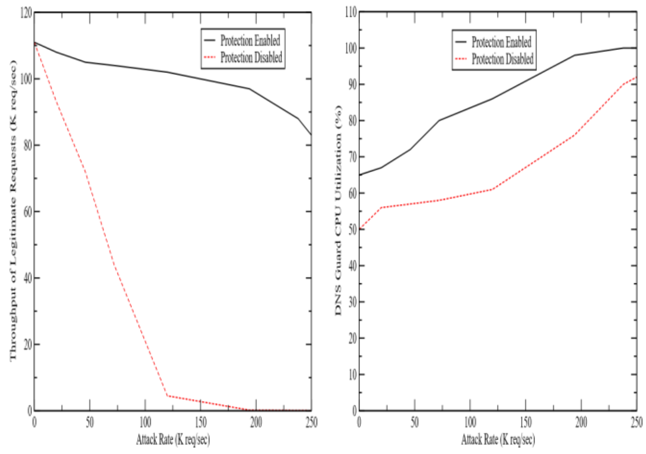
Then, we measure the throughput of the ANS when it is attacked. In this experiment, one ANS and three LRSs are used. The TTL of each DNS response is configured to be 0 to disable DNS caching. The DNS guard uses UDP based cookies with the first LRS, and TCP redirection with the second LRS. The third LRS sends UDP-based attack requests. By default the first and the second LRS each send UDP-based legitimate requests at a constant rate of 1K requests/sec. The sending rate of the third LRS is varied in the experiment. The DNS guard is configured to use the NS name mechanism for spoof detection. Other UDP-based spoof detection schemes show similar performance results.

Figure 4 shows the throughput of the ANS (the BIND server) and its CPU utilization when the DNS guard is turned on and off. When the DNS guard is completely turned off, the ANS's CPU utilization keeps on increasing with the attack request rate. As the attack request rate becomes greater than 12K requests/sec, the ANS starts to saturate and the throughput of legitimate requests drops dramatically, because the BIND-based LRS uses a large time-out value of 2 seconds. With a large time-out value, the legitimate request rate decreases very quickly even with very small loss rate. This test shows that BIND is extremely vulnerable to DoS attack. People may wonder why normal operation rarely shows this vulnerability. As reported in [22], the peak request rate at a root server is only 5 K request/sec, which is well below BIND's capacity.

Because spoof detection requires additional computation overhead, it is advisable to enable the DNS guard's spoof detection mechanism only when the input request rate exceeds a threshold. Since the ANS's capacity is around 14K requests/sec, we set the threshold at 14K requests/sec in this test. When the attack request rate is below 12K requests/sec, there is no spoof detection even if the DNS guard is enabled and all incoming packets go to the ANS. As soon as the attack request rate exceeds 12K requests/sec, the DNS guard's spoof detection mechanisms kick in, and the ANS's CPU utilization drops immediately because the DNS guard filters out all attack requests. At the same time, the DNS guard is able to maintain a fixed legitimate request throughput regardless of the increase in attack request rate. But the legitimate request suffers a little bit because the second LRS is redirected to use TCP and LRS's TCP maximum throughput is only 0.5K requests/sec.

3.4 DNS Guard Throughput under Attack

In this experiment, we use two LRSs, one as a legitimate LRS that already has the correct cookie for the ANS, and the other as an attacker that spoofs requests and does not have the right cookie. The legitimate LRS sends requests to the ANS as fast as possible. So the ANS is always saturated by the requests from the legitimate LRS. We then vary the attacker's DNS request rate to evaluate how effectively the DNS guard can prevent degradation of the legitimate LRS's received throughput in the presence of attacks.



(a) The throughput of legitimate requests. (b)

Figure 5: The modified DNS scheme and the DNS-based schemes

The modified DNS scheme and the DNS-based schemes can protect the throughput of legitimate DNS requests at a CPU overhead of 15% to 25%.

Figure 5(a) shows the throughput of legitimate requests. When the DNS guard is disabled, the throughput of legitimate requests decreases almost linearly with the increase in the attack rate, because legitimate requests can only take the remaining ANS capacity left by the attack requests. The reason for this behavior is that the legitimate LRS waits for 10 msec if requests are lost, whereas the attacker LRS never waits. When the attack request rate reaches around 110K requests/sec, the legitimate LRS is effectively halted. In practice, BIND use 2 seconds for the wait timer, which leaves legitimate LRSs at a even more vulnerable position.

When the DNS guard is enabled, the throughput of legitimate requests also decreases with the increase in attack request rate, but at a much slower rate. Consequently, the DNS guard is able to maintain the legitimate request throughput at no less than 100K requests/sec even when the attack request rate reaches 200K requests/sec. The reason for this graceful degradation behavior is that the DNS guard can recognize and drop attack requests so that they can never reach the ANS. That is, legitimate requests don't need to compete for the ANS's capacity with attack requests.

4. CONCLUSION

The key contribution of this research is that it provides a comprehensive study on the use of cookies in DNS spoof detection. Each DNS requester needs to obtain a unique cookie from a guarded ANS and accompanies all subsequent requests to the ANS with the corresponding cookie. Spoofed requests cannot present correct cookie thus can be detected. This seminar designed and implemented three schemes to embed cookies to DNS. The DNS-based scheme exploits mechanisms in the current DNS protocol by embedding cookies in NS name and NS IP address. The TCP-based scheme redirects LRS to use TCP and uses TCP sequence number as cookies. A kernel-level transparent TCP proxy is proposed to offload ANS from processing TCP connections. The modified DNS scheme extends the DNS protocol with cookies and achieves optimal performance. The first two schemes only need to add a DNS guard at ANS side while the third one needs add DNS guards at both the ANS side and the LRS side. These schemes are implemented as a Linux kernel module called DNS guard. The measurements on the DNS guard prototype demonstrate that the DNS guard can indeed protect ANS from DoS attacks by maintaining 80K requests/sec throughput in the presence of 250K requests/sec attacks. The TCP-based scheme can achieve 22K and 10K requests/sec throughput in normal operation and under attack, respectively. None of the three spoof detection schemes generates false positives, and there is no obvious way to evade the detection either. Finally the DNS guard can be deployed incrementally and transparently as traditional firewall.

REFERENCES

[1] Detecting and Preventing IP-spoofed Distributed DoS Attacks, Yao Chen1, Shantanu Das1, Pulak Dhar2, Abdulmotaleb El Saddik1, and Amiya Nayak1(Corresponding author: Shantanu Das)

[2] Detecting Spoofed Packets, Steven J. Templeton, Karl E. Levitt Department of Computer Science U.C. Davis

[3] Detecting DDoS Attacks Against DNS Servers Using Time Series Analysis Tong Guang NI, Xiao Qing GU*, Hong Yuan WANG

[4] Detecting and Preventing IP-spoofed Distributed DoS Attacks ARTICLE in INTERNATIONAL JOURNAL OF NETWORK SECURITY · JANUARY 2008

[5] Defense against DDoS Attacks Using IP Address Spoofing Archana .S. Pimpalkar1, A. R. Bhagat Patil2 PG Student, Department of Computer Technology, Yeshwantrao Chavan College of Engineering,

[6]http://www.webopedia.com/TERM/D/DoS_attack.html

[7] http://www.secure64.com/dns-guard