SSL/TLS Security Negotiation: Through Fake Certificate Injection

[#]First Author1, Second Author*2
[1]Abdul Jabbar Khilji
Research Scholar
JJT University, Jhunjhnu(Raj.)
[2]Mukesh M. Joshi
Associate Professor
Govt. College of Engineering and Technology, Bikaner

***Abstract*- When sensitive information must be sent through an insecure network like the Internet, one of the most important things is to encrypt the data ensuring confidentiality, data reliability, avoid data tempering and ensure non-repudiation. SSL is the technology that protects all that data and makes it safe to use untrusted networks. A form of encryption and potential authentication, SSL ensures that data remains in the middle attacks to establish a baseline level of trust between client and server. The paper presented the latest IE vulnerabilities and diffusion of worms, and so on.**

I. INTRODUCTION

Although both secure socket layer (SSL)[1] and Transport Layer Security (TLS) [2] are cryptographically secure as always, something utilizable exists. This is due to the obscurity encountered by users in understanding the public key infrastructure and its layered organization. Basically the end-user who wants to establish a secure connection to a server through SSL/TLS must first accept and trust the digital certificate that identifies the server. This ensures that the server is right that it claims to be. The end-user must trust the authenticity of the certificate. Thus, if we inject a fake certificate into the communication stream and the user trusts that the certificate is good, then the game is over. Know the algorithms used to encrypt the traffic and manage the public keys, having the certificate and the encrypted traffic; we can easily decrypt the whole traffic, allowing us to take out all sorts of information. If we exclude the highly impossible or more difficult methods of stealing the certificate, such as hacking the server or social engineering, we are still left with one more method. We may not have the real certificate but we have our own certificate. We create a fake certificate as similar as possible to the original one that we will inject into the communication stream and that will hopefully be trusted by the victim. This kind of extremely effective attack is named MITM (Man- in the Middle Attack) [3].

II. ATTACK ANALYSIS

The ways to attack this failing are similar; the only differ depending on our position in regards to the victim. Are we on the same subnet as the victim? Or are we on a different subnet? The first case is the simplest, the second one (if we cannot reach the victim directly on the LAN) is more complex but not impossible. In both cases, the victim must trust our fake certificate and start the communication with the other party without any uncertainties, in such a manner that we can collect all the encrypted traffic (encrypted with our certificate) so that it can be analyzed and decrypted later.

III. SAME SUBNET CIRCUMSTANCES

A. Assumptions

This paper assumes that the switches do not have a static map of the ARP entries and that nobody is using passive monitoring tools like arpwatch or Snort in-development processors to direct a suspicious address resolution protocol (ARP)[4] behavior. In fact, it has never been found that a network mapped with static ARP entries, and arpwatch is not used very often, for a lot of reasons (mainly simplicity and lazy administrators). There are of course proprietary solutions from Cisco [5], but they are more expensive.

B. Attack

This paper shows the ease of sniffing encrypted SSL/TLS traffic between hosts on the same network part, using very popular Open Source tools.

A Linux machine can be used to impersonate the attacker. The paper suggest that we may use virtual machines to test any type of hacking activity, including those confirmed here and we could also set up a test network to test the potential hacks. If we attempt the hacks on a live net-work, it is almost sure that we have a written agreement signed by the network administrator of the LAN we are hacking.

Phase 1:

Kernel IP forwarding

All of the packets that we want to sniff between Alice and Bob need to pass through us. In such a way the victims cannot recognize any suspicious activity. For instance, if Alice sends a packet to Bob, we will interrupt it, saving or modifying it, and then we need to forward it to Bob, otherwise it will cause a denial-of-service attack (DoS). The first thing we need to do is active kernel forwarding. To make it simple, we may use fragrouter [6], a tool written by Dug Song- a really useful one for a lot of purposes.

fragrouter-B1

Normal IP forwarding on Linux machines, we can do the same thing by changing the values of ip-forwarding, as following:

#echo 1 > /proc/sys/net/ipv4/ip_forward

Phase 2:

ARP Spoofing

Now, when we can forward traffic, we must be able to constantly spoof ARP-replies back to the victim in order to convince the machine that we are the second party in the communication process. We can do that easily with another sniff suit tool called arpspoof[7].

We specify the IP of the victim as a target and the IP of the machine wishing to intercept packet for – as a host. Usually, this machine is the local gateway (During the investigation we put the IP of the local DNS server in the example so that investigators do not create too much traffic).

#arpspoof –t 10.10.69.135 10.10.84.10

0:c29:6e:c5:8b 0:13:d4:bf:6f:

50 0806 42 :arp reply 10.10.84.10

Is at 0: c: 29:6e:c5:8b

Phase 3:

Listening

Now we have to prepare a trap to catch the victim's traffic when it starts an HTTPS session. We must listen and log three types of traffic, with three different programs, taking three different approaches. First, we shall look for DNS queries from the victim and intercept them. Another tool that for this task. Note that the attacker's IP 10.10.68.137 will be excluded by the program.

#dnsspoof

dnsspoof

dnsspoof: listening on eth0

[udp dst port 53 and not src 10.10.68.137]

Secondly, we need a tool to generate and automatically inject the fake certificate into the session. Webmitm[8] can help us with this. As stated in the man page, webmitm transparently proxies and sniffs HTTP / HTTPS traffic certificate, when the webmitm asks us whether it is the first time that we run it- it is the same generation process that we can find in Open SSL,, nessus-mkcert- we can run it in debug mode:

#webmitm-d

In this webmitm will listen for HTTPS connections, managing the whole communication for us with the client. After the previous setup is ~tqw~ completed, we must sniff all the plain/encrypted traffic with a common sniffer; tcpdump[9] is enough for our purposes for we do not need to view the connections in a real time.

#tcpdump-I eth0-w sniffed

tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes.

Now when the trap is prepared, we just have to wait.

Phase 4:

HTTPS Session Snaffing

We can recognize when the victim starts a HTTPS session because dnsspoof is mapping the queries and webmitm is injecting the certificate. The most exciting

part of this attack is just watching the output all the programs. After the trap is prepared, it is just a matter of waiting to collect all of the encrypted traffic.

Phase 5:

Decrypting sniffed Traffic

After the collection of the encrypted traffic we can stop listening and sniffing programs and start the decrypting activity. In order to do that, we need:

- Ssldump[10]- a tool to analyze encrypted SSL traffic.
- The tcpdump traffic dump- it contains all the traffic sniffed, including the encrypted ones.
- The certificate from webmitm that we have generated, we can decrypt cipher text knowing the key and the algorithm. We can also decrypt the SSL traffic knowing the certificate used in the session (key) and the protocol SSL (algorithm).

  #ssldump-r output_wireshark

-k webmitm.crt-d > decrypted output opening the output file with an editor, we can see all the decrypted sensitive data that was in transit under SSL. The practical use of it is grapping the useful information, like names of variables, in order to know their values. Imagine, we sniffed HTTPS traffic with authentication pages and now we are searching for web from user-name/passwords. To be able to grep the correct variable names, we must see the source code and the layout of the web page online. Firefox is quite useful for this especially with extension like firebugs[11] that can help us to see the divs and the frames that will let us know the name of the variable in which the suer/password will be stored. The paper mentions the Firefox plug-ins because sometimes it is very difficult to grep the useful information. We may think that the name of the variable is one thing, but in fact it is different. For example, a bank account website is tried. It is really secure website, written in JEE with JSP. The login page is a mixture of multiple JSP frames and div. the code is then make grep in the ssldump output file. In the second case, where the attacker and the victim are on different subnets, the though becomes more complex. The good thing is that even though we need new techniques to route the victim traffic to when spoofing the certificate, sniffing and dumping the SSL traffic, are still valid. And, of course, we need new techniques because we cannot ~tqw~ use the classic arp spoofing solutions when the victim is on a different subnet. There are different techniques that we can use to obtain the results in this particularly difficult situation, and all of them are based on a direct/route of the traffic from the victim to us. The paper excludes some of the techniques. Not because they are so difficult to succeed if the network environment is hard ended, and/or we are not really skilled.

Routing Protocols

This paragraph is about breaking into a router, changing its configuration and adding a static route or a GRE tunnel to our machine. The game is over if we can find a way to compromise the router, finding bugs, guessing SNMP[12] community strings, brute-forcing password hashes and doing social engineering. This paper also presents route mangling: the protocols involved here different (IGRP, OSPF, RIP, BGP), and just a few implement cryptographic authentication of the routes with MD5. For this reason there is a large number of tools that can exploit these protocol weakness, like technique used by hackers to make MITM attacks successful if they are geographically dislocated from the victim, is the famous DNS cache poisoning attack. There are two ways to alter the DNS cache with our data:

- Negotiation the DNS Server
- Exploiting a weakness on the DNS server implementation or the protocol itself

The first way is the most effective but hard to accomplish, depending on the hardening of the DNS server. If for instance DNS is running on a hardened version of Linux with RSBAC patches configured properly, and if djbdns was the choice of the administrator, then the task of exploiting the server becomes really hard. If some bugs are discovered in DNS software, this attack becomes possible then. Just think about the latest Microsoft DNS server 0-day exploit published in Milw0rm. Usually the preferred way is to use some form of a known trick to force the protocol to do what we want. All of these known tricks are based on bad server implementations. For example, some years ago version 4 and 8 of the BIND DNS server, were vulnerable to 2 types of attacks known as DNS ID spoofing and Birthday Attack. Both are based on the same weakness. The exploitable bug was caused

by a bad implementation of random ID generator for the lookup process.

If the lookup id based on UDP packets just 2 bytes are dedicated to it, meaning that there exist only 65535 possible ids. Guessing the nest Id or by flooding the server with spoofed request, it was possible to alter the cache of the DNS server. Strictly related to the Birthday paradox taking help drastically reduce the number of spoofed UDP guesses and thus preventing the flooding. If the paper attracts some attention, a closer look is suggested at two more famous programs that will help us with DNS poisoning: ADMID pack and Zodiac. They are really useful if we wish to understand in depth the attacks that the paper has mentioned. Also there are patches for those vulnerable BIND versions. A lot of servers are still exploitable because they have not been patched. It is just a matter of finding them.

IV. LIMITATIONS OF SSL: LACK OF STANDARDS

The SSL protocol is well designed with respect preventing eavesdropping and avoiding successful man in the middle attacks. It is less concerned with the processes and procedures that a person or organization must go through to acquire a certificate.

Lack of Authentication Standards

The SSL protocol depends on the existence of a trusted third party. It is assumed that parties that want to communicate over a secure channel can agree on an organization that will vouch for the identity of holders of SSL certificate:

- What constitutes sufficient proof of identity?
- Are there varying levels of proof?
- If so, how will certificates represent the varying levels of proof?
- How can one be sure that different CAs follows the same standards for identifying a party?

These issues all move us from the realm of cryptography and network protocols into the often more complex organizational and procedural issues that surround CAs.

Varying Levels of Certification

Rarely in business or government operations is there a situations in which one size fits all security are requirements especially variable. Consider a simple analogy with locks on doors. Sometimes a relatively inexpensive and weak lock is sufficient to meet one's needs, for example to keep a toddler from getting into a cabinet filled with chemical cleaners. It could invest in a stronger lock, but it would not add any advantages to the existing solution. An entire house however is likely to have stronger locks that will better protect its inhabitants and their possession. The additional cost and effort required to use the better locks is well justified. Finally a bank an obvious target for thieves, will use specialized locks and additional security measures to protect its assets.

Domain-Only Certificate

In the case of online transaction, different needs dictate different levels of security and authentication. As an example, a web master running a site for a local basketball league wants to allow coaches to use the site to post practice schedules and other team-related information. The web master does not want anyone else changing those schedules, a user login is implemented. Being security conscious, it is not desired that clear text passwords be sent over the internet, so clients and the web server. This is a relatively low-security environment. There are no financial transactions, no exchange of confidential personal information, and no potential for significant loss of intellectual property. Coaches, if they are concerned at all about submitting their usernames and passwords, would likely want nothing more than to be assured that the transaction is encrypted. In this case, simply having a certificate that verifies the identity of the domain is sufficient. Domain-only certificate typically validate that the requestor of a certificate is authorized to use that domain. These certificates are inexpensive, largely because the validation process can be automated. Information about the owners of domain names is

readily available from utility programs, such as whose.

Domains-only certificates have lowered the cost of using SSL, which has been a benefit to many; unfortunately, they have also lowered the cost of starting phishing sites that look legitimate. They have also led some companies to use lower grade certificates rather than authenticated certificates to protect sensitive data. More extensive authentication procedures should be used for most business-oriented domains.

Full-Company Validation

When CAs uses full validation procedures, they look for more rigorous proof of the identity of the person, business, or organization requesting certificates. They will still go through the same steps as the domain-only validation, but in addition they will do things such as:

- Verify the existence of a physical address of the person, company or organization.
- Check government records to verify a business is legally established.
- Require copies of documentation, such as driver's license for a person or incorporation papers for company with full-company validation, one cannot simply register a domain name and require a certificate; the requester must be able to demonstrate the company has some established legal identity. Again, there are verifying levels of certification involved depending on the issuing CA.

Problems with Varying Levels of Certification

The biggest problem with varying levels of certification is that these variances are not apparent to users who are expected to trust these certificates. When a browser establishes a SSL session with a server, the same lock icon will appear on the browser whether the server certificate is domain-only of full company validation. A phishing site can look as legitimate as a real bank's site. The root of this problem was there no well defined standards foe authenticating businesses. Two different CAs may have different procedures for full company certification. One company may check government records to see if business by a certain name has been established while another will make more rigorous checks to see that company is actually still actively in business. These variations in current practices, along with the rise of phishing scams, have undermined trust in online commerce and prompted the industry to respond with a new type of SSL certificate that does not suffer from these deficiencies.

EV SSL Certificates

EV SSL certificates use the same cryptography and network protocols as SSL certificates but they improve the certification process to address the weakness outlined earlier. The standards for EV SSL certificates have been established by a governing body known as the CA/Browser forum. Before an EV SSL certificates is issued, the CA conducts a through and standardized process to verify the identity of the requestor. The steps include:

- Verifying the entity physically exist.
- The entity is legally recognized.
- The entity is actively conducting business or other operations
- The identity of the entity matches the identity on legal records
- The entity has legitimate use of the domain
- The individual requesting the certificate is an authorized representative of the company in question.

CAs that issue EV SSL certificates also subject to audits performed by Web Trust. A professional assurances organization, to demonstrate that proper policies, procedures and training measures are in place to ensure quality control. In addition most high-security browser such as Microsoft IE7 now provide

additional visual cues to user when a site how a user is to know the level of verification and authenticate behind a certificate.

V. CONCLUSION

We cannot always trust even the most secure solution especially if it is trusted by the human factor. SSL and TLS add a layer of security to our communication, but they have to be understood in depth. Too many times the most effective attacks can be launched if the victim is doing what we want. We must think about the attacks that have been discusses plus the latest IE vulnerabilities discovered by HD Moore, the diffusion of worms and so on. It can become uncontrollable if we are a security manager of an enterprise and our employees don't follow the specified security policies.

REFERENCES

[1] http://www.webopedia.com/terms/S/SSL/Html.

[2] http://searchsecurity.techtarget.com/sdefinition/0,sidl4_gci557332,00.htm

[3] http://it.toolbox.com/wiki/index.php/man-in-the-middle-attack

[4] http://linux-ip.net/html/ether-arp.html

[5] http://www.ciso.com

[6] http://www.rpmfind.net/linux/rpm2/search.php?query=fragrouter

[7] http://su2.infdo/doc/arpspoof.php

[8] http://irongeek.com/i.php?page=backtrack-3man/webmtim

[9] http://www.tcpdump.org

[10] http:// www.rtfm.com/ssldump

[11] http://addons.mozila.org/en-US/firefox/addon/1843

[12] http://www.dpstele.com/layers/12/snmp_12_tut_part1.php

[13] http://www.phenoelit-us.org/irpas/

[14] http://namesis.sourceforge.net/