The most recent SSL security attacks: origins, implementation, evaluation, and suggested countermeasures

Wassim El-Hajj*

Computer Science Department, American University of Beirut, Beirut 1107-2020, Lebanon

ABSTRACT

Attacks have been targeting secure socket layer (SSL) from the time it was created especially because of its utmost importance in securing Web transactions. These attacks are either attacks exploiting vulnerabilities in the SSL protocol itself, or attacks exploiting vulnerabilities in the services that SSL uses, such as certificates and web browsers. While the attacks on SSL itself have been successful, at least in the context of academics or other research, attacks on the services that SSL uses have been successfully exploited in an actual commercial setting; the fact that makes these kinds of attacks extremely dangerous. In this paper, we give a brief overview of the attacks conducted on the implementation of SSL and we analyze in more details the recent attacks that exploit the services SSL uses. Most of these attacks are considered Man in the Middle (MitM) attacks. In particular, we explore the most recent five attacks targeting SSL: SSL sniffing, MD5 collide certificate, SSL striping, SSL Null prefix and online certificate status protocol (OCSP) attack. We discuss the origins of each attack and explain the typical environment that allows for such attacks to occur. We then highlight the implementation phase where we implemented some of the attacks and were able to catch logins, passwords, and any data transmitted between two parties. In addition, we implemented using Java, our own parsers and decoders to extract the useful data from the captured files and decode them if needed. Since most of the discussed attacks target browsers and the way they manage certificates, we conducted an extensive evaluation on the rate of success of the SSL attacks when various browsers are used. The browsers that were considered are Internet explorer (IE), Mozilla Firefox, Opera, Safari, and Chrome. The alarming results show that all analyzed attacks except for SSL Sniffing can be performed on almost all browsers. Copyright © 2011 John Wiley & Sons, Ltd.

KEYWORDS
secure socket layer; man in the middle attacks; certificates

*Correspondence
Wassim El-Hajj, Computer Science Department, American University of Beirut, Beirut 1107-2020, Lebanon.
E-mail: we07@aub.edu.lb

1. INTRODUCTION AND BACKGROUND

With the great evolvement of the Internet, people started to realize the importance of securing sensitive data that is transmitted over the public Internet, such as credit card information, usernames, and passwords. Consequently, many security protocols have been proposed in order to address the security flaws and vulnerabilities that can be or have been exploited by attackers. One of these protocols that is being used extensively nowadays is secure socket layer (SSL), SSL, and under its new name transport layer security (TLS), is one of the most important and well known protocols that achieve data confidentiality, integrity and authentication in Web transactions. It works between the OSI Application and Transport layers to provide for the client and server applicable connection oriented mechanisms for secure communication channel. Even with the presence of some attacks that are able to extract various transmitted secure credentials, SSL has been increasingly used since its introduction. Some specialists even argue that SSL is partly responsible for the emergence of e-commerce and other security sensitive services over the web.

Unfortunately, many attacks have been successfully conducted on SSL. These attacks can be categorized into (1) attacks that exploit the implementation of SSL itself, and (2) attacks that exploit the vulnerabilities in the services and tools that SSL uses. In the rest of this section, we start by giving a brief overview of some important
attacks following category 1 (first type of attacks). We then
describe two old but rather important Man in the Middle
(MitM) attacks following category 2 (second type of attacks):
SSL MitM attack against Internet explorer (IE)
secure objects and Honeypot SSL MitM attack. In later
sections of the paper, we consider the most five recent SSL
attacks following category 2. We discuss, implement,
evaluate, and suggest counter measures for them.

In Ref. [1], a remote timing attack is described where the
attacker observes the time taken to execute specific
cryptographic parameters. The recorded times are then
used to compromise SSL security. In Ref. [2], an attack,
called Truncation attack, is described where the attacker
prematurely closes the SSL session between two users.
This attack is carried out by sending the SSL “close_notify”
message on behalf of one of the peers. The other peer
cannot tell if it was a legitimate end of data or not. In
Ref. [3], researchers identified a practical way to launch
denial of service (DoS) attacks against TLS/SSL. The
attacker starts by intercepting the messages exchanged
between the client and the server. Afterwards, client
requests are either dropped or responded to through the
server with some error messages causing the handshake
authentication phase to fail. In Ref. [4], two different attacks
are presented: Cipher Suite rollback and Change Cipher Spec
(CCS) dropping. The Cipher Suite rollback is an SSL v2.0
attack in which the attacker forces the client to roll back to
a weaker group of encryption algorithms, thus making the
secret communication vulnerable to decryption attacks. CCS
dropping attack works by intercepting the legitimate SSL
communication and dropping or deleting the CCS message.
This causes the two involved parties never to update their
Cipher Suites leading to unprotected communication.

Although the above presented category 1 attacks are
dangerous, some careful analysis will show that most of
the successful commercial attacks on SSL have nothing to
do with the SSL protocol implementation itself, but rather
with the deployment of the protocol over the web (category
2 attacks). Most of these successful attacks are mainly
based on the MitM attack where the attacker places its
machine between the client wishing to access a secure
website and the server.

SSL MitM attack against IE secure objects was launched
in 2001 against Microsoft IE 5.0/5.5/6.0 [5,6]. It exploits a
flaw in the way IE handles https objects, specifically image
objects. The basis of this attack comes from the fact that older IE versions, 5.0/5.5/ and 6.0, only check if
the server certificate was signed by a root certificate
authority (CA); they do not perform any further checking
such as the expiration date or the common name in the
certificate. The attacker uses a MitM tool to place itself
between the victim and the server. When the victim tries to
access the server (let us say www.serverSite.com), the
attacker uses an HTTP attacking tool to append a 1 × 1
pixel image into the html page that is returned to the
victim’s browser. Since the image does not exist, it will appear in the victim’s browser
as an empty pixel, which cannot be noticed. The attacker
also sends its own certificate, valid or stolen, to the victim’s
browser. In turn, the victim’s IE attempts to fetch the file
“nonexistent.gif” over https and thus it caches the bogus
certificate in association with www.serverSite.com, and
flags the certificate as trusted for the remainder of the
browser session. When IE uses https to fetch other objects
from www.serverSite.com, it uses the cached bogus
certificate without any further checks. The victim can
catch this attack only by noticing the empty image or
manually checking the servername in the certificate.

Honeypot SSL MitM attack [7] uses a Wi-Fi attacker
called Honeypot that enforces clients to connect to a
Honeypot access point (AP) instead of the legitimate one.
While the client is searching for legitimate APs to connect
to, the attacker changes its service set identifier (SSID) to
be the same as the SSID of the legitimate AP. This attack
usually starts at layer 2 of the OSI model, but it was
tweaked to launch a MitM on SSL. As soon as the attacker
succeeds in getting a layer 2 association with the client, the
wireless interface can be bridged with the Internet
connectivity interface. By doing so, the attacker will have
IP layer connectivity with the client and thus it can act as
MitM between the client and the server. Unlike previous
MitM techniques which require address resolution protocol
(ARP) cache poisoning for traffic redirection, Honeypot
attacker is much easier since all the traffic passes through
the Honeypot with successful client association. Proxy
server with domain name server (DNS) spoofing is still
required. The former is needed to pass the traffic to the
intended website, while the latter is needed to make sure
that all traffic goes through Honeypot attacker. As any
normal SSL MitM attack, the Honeypot attacker needs to
introduce faked certificate to the client to achieve faked
secure connections. To make this attack more robust, tools
such as hotspotter [8], Karma [9], and essid-jack [10] were
developed to launch the Honeypot attacker. Clients can
protect themselves from this kind of attack by turning-off
their wireless interfaces when idle; deleting any unsecure
or weakly secured wireless profiles, updating their OS with
any new released security patches, refusing any untrusted
certificate, and finally enabling server certificate validation
via proper client’s configuration.

The rest of the paper is organized as follows: In Section
2, we give a brief overview of SSL. In Section 3, we focus
on analyzing the most recent five attacks targeting SSL:
SSL sniffing, MD5 collide certificate, SSL striping, SSL
Null prefix, and online certificate status protocol (OCSP)
attack. All of these attacks are related to each other one way
or another. In reality, most of them are MitM based attacks.
We consider each attack at a time and discuss its origins
and the vulnerabilities it exploits to conduct the attack. We
also discuss the methodology used to carry out the attacks
and consequently we implement some of them. Since most
of the discussed attacks target browsers and the way they
manage certificates, we include in the paper an extensive
evaluation on the rate of success of the discussed SSL
attacks when various browsers are used. The browsers that
were considered are IE, Mozilla Firefox versions 3, and 3.5, Opera pre-v8 and post-v8, Safari, and Chrome. In Section 4, we conclude the paper and summarize our contributions.

2. BRIEF OVERVIEW OF SSL

When someone tries to access a secure website such as Gmail or Paypal, some sensitive information, for instance passwords, might need to be exchanged between the client and server. During the exchange of such secure information, http is changed to https indicating that the communication is being transmitted using HTTP over SSL.

SSL is a cryptographic protocol originally founded by Netscape. It is integrated as part of many browsers such as IE, Mozilla Firefox, Opera, Safari, Chrome, and most Web server products. SSL provides confidentiality, data integrity and authentication for communications over TCP/IP networks. It is supposed to make an unsecure communication secure, such as secure communication over the Internet which runs on behalf of higher-level protocols such as HTTP. As a result, SSL permits an SSL-enabled server to authenticate itself to an SSL-enabled client and vice versa (optional). After the negotiation on the cryptographic algorithms to be used, an encrypted tunnel is established between the two parties. Detailed analysis of SSL can be found in Ref. [11]. SSL provides the following services.

2.1. SSL server authentication

This property allows a user to verify the server’s identity. SSL-enabled client software can use standard techniques of public-key cryptography to check that the server’s certificate and public ID are valid. Then, it checks whether it has been issued by a trusted CA listed in the client’s list of trusted CAs. This verification might be critical if the user is sending sensitive information such as credit card details over the network and he wants to make sure that the server is really who he claims to be. Figure 1 shows the four checks conducted by the client to authenticate the server.

2.2. SSL client authentication

The SSL protocol also allows a server to verify the client’s identity by using the same method discussed above reversed. This kind of client authentication is usually optional in a client-to-business environment but mandatory in a business-to-business environment. The SSL-enabled server software can confirm whether a client’s certificate and public ID are valid. Also, it checks if it has been issued by a trusted CA listed in the server’s list of trusted CAs. For the client, this verification might be critical if the server, say a bank server, is sending confidential financial information to a customer and wants to make sure that the client is who he claims to be. Figure 2 shows the five steps conducted by the server to authenticate the client.

2.3. Encrypted SSL connection

SSL encrypts all information sent between a client and server using symmetric encryption. Before that, both parties have to agree on the cryptographic algorithms (Cipher Suite) that should be used in the secure communication for the purpose of achieving confidentiality, integrity, and authentication. Since the client initiates the communication, it has the responsibility of proposing a set of Cipher Suites to be used in the exchange. The server selects from the client’s proposed suites indicating the cryptographic algorithms the two systems will be actually using. An example of such a Cipher Suite is: TLS_RSA_WITH_RC4_128_MD5, which means that the protocol used is TLS, the public-key algorithm is RSA, the...
The symmetric cryptographic algorithm is RC4 with 128-bit keys, and the message MAC is HMAC_MD5. Although the final decision rests with the server, the server can only choose from among those options that the client originally proposed. After agreeing on the Cipher Suite to be used, the client creates a random pre-master key and sends it to the server encrypted with the server’s public key. When the server receives the message, it extracts the pre-master key by decrypting the message with its private key. The pre-master key is then used by both parties to create what is called a Master Secret that is used afterwards to create all the needed session keys. From this point on, symmetric key cryptography will be used for the actual encryption of the transmitted data between the client and server. When the secure communication halts, SSL closes the session. More details can be found in Ref. [11].

3. SSL RECENT ATTACKS

Most of the commercial attacks over SSL are based on the exploitation of design vulnerabilities. In most occasions, it is not the security protocol itself, but rather the improper design of trust models and web browsers that constitute the main sources of dangerous attacks; attacks such as MitM. During the past few years, different attacks were conducted against SSL. Most of them were targeting the design vulnerabilities of the clients’ browsers deploying SSL technology. It is to be noted that different Web Browser companies have different implementations of SSL. In what follows, we explore the most recent five attacks targeting SSL: SSL sniffing attack, MD5 collide certificate, SSL striping attack, SSL Null prefix attack, and finally OCSP attack. These attacks share many similarities: (1) most of them are based on MitM attack and (2) all of them except SSL stripping use a tool called sslsniff [12]. Now, there is a newer more powerful version of the sslsniff tool called sslsniff v6.0.

3.1. SSL sniffing attack

3.1.1. Origin.

One Web Browser vulnerability that can be exploited by attackers is related to a field in the certificate called “Basic Constraint”. Given a certain certificate, an example of the Basic Constraint field is a parameter named path length that indicates the maximum number of CA certificates above this one in a certification path. This path length constraint is used to ensure that the holder of the certificate can only issue end entity certificates, not CA certificates. If the browser did not check this field when it comes to validating a certificate, an attacker can use this certificate to sign a request of a forged certificate. This forged certificate...
will be treated by Web Browsers as a trusted certificate. When no basic “constraint field” validation is done, clients' browsers will consider the chain in the forged certificate as a trusted chain, and hence establish a secure channel with attackers rather than legitimate entities.

3.1.2. Attack scenario.
Basically, ssldsniff version 5 was developed to exploit the vulnerability in Microsoft IE that allows leaf certificates to sign other certificates. This vulnerability is originated as a result of bypassing the “Basic Constraint” field validation. The attack is based on the MitM technique where an ARP spoofing tool called arpspoof is used to achieve traffic redirection. As a result of using arpspoof, all the traffic between the client and server will be redirected to the attacker machine instead. Figure 3 depicts the details of the SSL sniffing attack. When the real server replies back to the real client providing it with its certificate, the attacker intercepts this message taking by this action the client’s identity. At this point, the attacker has a secure communication with the server. Now the attacker creates a certificate on the fly (discussed below) imitating the server (a certificate that includes the server information) and signs it with whatever valid certificate it possesses. This newly created and signed certificate is valid because the certificate chaining leads to a valid authentic CA. This certificate is then sent to a client resulting in another secure channel between the client and the attacker. Using the MitM attack and on-the-fly certificate generation, the attacker will open two secure channels, one with the victim, and the other with the legitimate server. At this point all the secure communication is intercepted by the attacker.

To implement this attack, we used ssldsniff and arpspoof tools. Any password parser to retrieve the captured credentials from the bulk of data could be used. We performed the attack under Linux. Note that, performing the steps of this attack might differ from one environment to another. These steps are as follows:

1. Enable the promiscuous mode to successfully receive traffic from the victim and then forward them back to the designated server. As a root, the sysctl.conf configuration file should be modified by Un-hashing the following command: net.ipv4.ip_forward = 1.
2. Redirect the entire victim’s http traffic to the attacker port which is 10000 (ssldsniff default port). As a root, type the following command: iptables-t nat-A PRE-ROUTING-p tcp-destination-port 80-j REDIRECT-to-port <ListeningPort>.
3. Corrupt the victim ARP cache table so that all the traffic will be redirected to the attacker instead of the gateway. As such, you need to know the IP addresses of the victim and his gateway. The command below can be used: arpspoof -i <NetworkdDevice> -t <Target> <RoutersIpAddress>.
4. Start ssldsniff tool which takes as input the forged certificate and the location of the output log file. The following command should be used: ssldsniff -p 10000-c trust.crt-d/√~.

To get the credentials of the victim, you can either use a password parser to extract the credentials directly from the file, or search the file manually.
3.1.4. Evaluation.
According to our study, Microsoft IE versions prior to version 7 are all vulnerable to the SSL Sniffing attack (Figure 4). IE used to not check the Basic Constraint filed at all. Therefore, an attacker can buy a legitimate certificate and then use it to sign the certificate that will be created on the fly. When this attack happens, the user will notice no difference at all in the address bar and no security warning messages will be displayed. When this attack was disclosed in public, Microsoft directly tended to fix the problem by disallowing leaf certificates from signing other certificates. Therefore, this attack is not possible anymore.

However, the sslniff tool is still useful. Using it with invalid chaining certificates with any browser will only display a security warning message to the client warning him that the certificate has not been verified by a trusted issuer; but the access to the site will still be possible by adding an exception. Such an attack is still dangerous because unknowledgeable users that have no background knowledge about certificates will blindly accept the warning message and proceed by adding the exception.

Recall that this attack was feasible because of a vulnerability that allowed leaf certificates to sign other certificates. In the next section, we investigate an attack close to this one but is based on creating Rogue CA certificates from scratch.

3.2. MD5 collide certificate

3.2.1. Attack origin.
“Computational Infeasible” is a main security requirement of cryptographic algorithms. However, cryptographic algorithms that were computationally infeasible to crack some decades ago could be cracked nowadays using the recent advancements in technology. MD5, a widely used cryptographic hash function, has been employed in a wide variety of security applications. It is supposed to provide the following properties:

- Given message \( m \), it is computationally easy to get the hash \( h = H(m) \).
- Given hash \( h \), it is computationally infeasible to get \( m \) out of \( h \), such that \( h = H(m) \).

- Given \( m \) and \( h = H(m) \), it is computationally infeasible to get another message \( n \), such that \( H(n) = H(m) \). This property is known as collision resistance.

However, it has been shown that MD5 is not collision resistant; as such, MD5 is not suitable for applications like SSL certificates or digital signatures that rely on this property [13]. Unfortunately, some CAs still use MD5 in SSL certificates and digital signatures, and hence, compromising the security of their clients. Despite the warnings made in 2004 regarding the danger of MD5 collisions, statistics in 2008 showed that 9000 certificates are still using MD5 rather than SHA1.

3.2.2. Attack scenario.
On 30 December 2008, a group of researchers were able to generate two different certificates both having the same certificate digests (signatures) [14]. This allows a malicious user to create a rogue CA certificate trusted by all major browsers and other common applications. Any certificate for any domain that would be signed with this rogue CA would be trusted and accepted, without any form of warning, by all major browsers. In particular, this opens the door for virtually undetectable phishing and manipulation attacks on any secure webpage [15].

To start the attack, the attacker needs a valid legitimate certificate that is signed with a real CA. The certificate should use MD5 and should be accepted by all browsers (let us call this certificate \( \text{real.cert} \)). One root CA that provides these properties is Equifax Secure Global eBusiness CA-1. Next, the attacker issues a certificate request to be signed by this CA. As expected, the Equifax authority will sign the certificate request and send it back to the client. At this point, the attacker creates another certificate request with crafted information that will help to cause the MD5 collision. However, he does not send the request to a legitimate CA, but rather he uses the same signature digest of the first legitimate certificate (\( \text{real.cert} \)) and add it to this rogue certificate (say \( \text{rogue.ca.cert} \)). This rogue certificate is valid over the signature. Anyone who examines the rogue certificate will think that it has been signed by Equifax authority while the fact is that Equifax has never seen this certificate before.

Now, the attacker is in possession of a rogue certificate that can be used to sign other certificates. Since the rogue certificate has been created by the attacker and not by a CA, the attacker sets the Basic Constraints field to TRUE. This means that the attacker is able to use this certificate to sign other leaf certificates. From now on, the attacking scenario will be the same as the scenario described in Section 3.1. The researchers in Ref. [14] used sslniff version 5 tool to conduct the attack. Figure 5 presents the details of the attack.

3.2.3. Attack evaluation.
As long as Internet browsers are still accepting MD5 hash function, all these major browsers are vulnerable to this attack. Even though this vulnerability is targeting SSL, but
it has nothing to do with the SSL protocol implementation. The weakness came from the fact that companies kept using MD5 weak algorithm despite the warnings, and refused to use stronger hash functions, such as SHA1 and SHA2. Moreover, newer versions of IE have not revoked those kinds of certificates yet. As shown in Figure 6, our evaluation study shows that many of the current browsers are still vulnerable to this attack.

The best defense against this attack is to start using SHA1 or even switch to SHA2 because it will not be long till someone finds a way to attack SHA1. We were not able to implement this attack because we do not have enough computational power to create a forged certificate. In the next section, we analyze and implement another recent dangerous SSL attack called SSL Stripping.

3.3. SSL stripping attack

3.3.1. Attack origin.

Usually, websites do not use SSL connection throughout the opened session, but rather they only use it when sending sensitive data over the web. Since some URLs are too long, HTTP redirection is used to facilitate for the user the access to the long URLs through shorter ones. This is known as HTTP 301 permanent redirection [16]. If the URL is a secure one, another redirection known as 302 temporary redirection is used. Although this kind of implementation adds some user friendliness, it provides a vulnerability that the attackers can exploit to attack SSL; instead of redirecting the traffic to the secure server, the attacker can redirect the traffic to himself (MitM).

![Figure 5. MD5 collision attack.](image1)

![Figure 6. The browsers that are vulnerable to collide certificate attack.](image2)
3.3.2. Attack scenario.
When normal users want to access their Google e-mail accounts, for example, they would most likely type: www.gmail.com in the address bar. You can easily notice that the URL expands and becomes something like https://www.google.com/../../../accounts. It is extremely hard for users to memorize and type the long URL (https://www.google.com/../../../accounts) in the address bar; instead they can just write www.gmail.com. In the background, an HTTP GET request is sent to the server asking it to retrieve the location of the Google mail. The server replies with HTTP 301 status code with the location of Google mail. HTTP 301 status code means that the user connection (www.gmail.com) has been moved permanently to the new address which is: http://mail.google.com. When the user receives the new location, another HTTP GET request will be sent along with the new address location. However, this time the server will reply with an HTTP 302 status code. This status code means that the connection session has been moved temporarily to https://www.google.com/../../../accounts and is ready to accept user information in a secure manner. Now, the user’s browser will fetch the secure https website and send the credentials encrypted with the public key of mail.google.com.

As can be noticed, the users never access https secure websites directly; they actually access them through 302 redirections from normal http to https. This introduces a vulnerability that targets SSL but again has nothing to do with the protocol implementation itself. Figure 7 shows the attack that exploits such vulnerability.

This vulnerability can be exploited by fooling users and making them believe that they are really communicating with the legitimate server while they are actually communicating with a forged mimicked website. The attacker can exploit specifically the 302 redirections to let the secure traffic be redirected to his machine instead of forwarding it to the legitimate server. The point where the 302 redirection occurs is the point where an elusive tool called ssstrip [17] can be used to tell the users to navigate http://www.google.com/../../../accounts rather than https://www.google.com/../../../accounts. Notice that this new URL is http and not https. At this point, all the traffic exchanged between the client and the attacker is exchanged in the clear. However, the communication between the attacker and the server is completely legitimate and encrypted using the server’s legitimate public key. The server’s response will be decrypted by the attacker before forwarding it to the client.

3.3.3. Attack implementation.
To launch this attack, we used ssstrip and arpspoof. Any password parser to retrieve the captured credentials from the bulk of data can be used. The first three steps for conducting this attack are similar to the SSL sniffing attack (Section 3, Section 3.1). As for the last step, the command that should be used is: Python ssstrip – w secret.

3.3.4. Evaluation.
Aside from the HTTP technical vulnerability, this attack exploits people’s unawareness of secure communications and it works on all Internet browsers with no exception (Figure 8). All what a normal user wishes to have, is a fluent access to the designated website without paying attention to the address bar in the browser if it is http or https. However, avoiding 302 redirections can help protect clients against such kind of attacks. This can be done by
adding the https URL into the browser’s bookmark. So whenever a user would like to access the secure login webpage, it will be downloaded into the browser without 302 redirections.

3.4. SSL null-prefix attack

3.4.1. Attack origin.
Null-prefix attack [18] is another silent MitM attack that is based on exploiting two main aspects: (1) the treatment of the fields obtained from X509 certificates via most contemporary SSL/TLS implementations, and (2) the signing process of contemporary Certificate Authorities.

X509 certificates support PASCAL strings which are represented as a series of bytes mediated by another series of bytes indicating the length of the string followed by the data itself. C strings, on the other hand, are represented as a series of bytes terminated by the NULL character (’\0’). So, sending a request to Verisign to issue a certificate for ‘www.paypal.com\0.attacker.com’ is completely valid. Note that this string includes the C NULL character in the middle. This presents a vulnerability that constitutes the basis of the SSL null-prefix attack.

3.4.2. Attack scenario.
When a user submits a Certificate Signing Request to the Certificate Authorities, it validates the identity of the owner based on a comparison between the domain listed in the “common name” field retrieved from the request (www.paypal.com\0.attacker.com), and the root domain retrieved by looking up WHOIS database. The identity information is only associated with the root domain, so the CA does not care about the content of the any subdomains that might be present in the users’ requests. Consequently, submitting a request for www.attacker.com or www.paypal.com\0.attacker.com to Verisign does not yield any difference as long as it can prove that you are the owner of attacker.com.

Unfortunately, most contemporary SSL/TLS implementations uses ordinary C strings functions for manipulation and comparison and not PASCAL strings. A string comparison between www.paypal.com and www.paypal-.com\0.attacker.com will be identical. Therefore, the owner of www.paypal.com\0.attacker.com certificate can present his certificate for connections intended for www.paypal-.com, and thus breaking the authenticity of the intended server. Figure 9 shows the details of this attack.

3.4.3. Evaluation.
SSL null-prefix attack is possible because of weaknesses in Microsoft’s Crypto API or GnuTLS and flaws in how network security services (NSS) functions (described above). Among all evaluated browsers, Mozilla’s NSS is the worst because it requires the attacker only to obtain a single certificate with “common name” filed as “\0.attacker.com in order to intercept all the traffic.
initiated by NSS applications to any server. Figure 10 indicates the browsers that are vulnerable to this attack.

### 3.5. OCSP attack

#### 3.5.1. Attack origin.

Online certificate status protocol is designed to support clients’ real-time requests to check the validity of the certificates presented by the server. When a web browser, let us say IE is presented with a new certificate (one that it did not see before), it contacts the certificate issuer via OCSP to see if the certificate is still a valid one before accepting the certificate.

An attack can be conducted to make it very hard to revoke the Null-Prefix certificates described in the previous subsection [19]. It is a mode that can be activated when running sslniff tool. The attack is based on exploiting the OCSP response. The basic OCSP response structure is as follow:

\[
\text{OCSPResponse} ::= \text{SEQUENCE} \{
\quad \text{responseStatus OCSPResponseStatus,}
\quad \text{responseBytes [0] EXPLICIT ResponseBytes OPTIONAL}
\}
\]

Table 1 shows the components of the OCSP response in details. In the responseBytes structure, there exists a field that requires the signature of the CA; it is highlighted in bold. To forge the OCSP response, the attacker first needs to avoid the “successful” status under the responseStatus structure, since that would require including the CA’s signature in the responseBytes structure; a signature the attacker cannot obtain. Even though “malformedRequest”, “internalError”, “sigRequired”, and “unauthorized” status do not require a signature, the attacker needs to eliminate them because they convey error conditions. However, the “tryLater” status sounds good since is does not require a signature and it does not convey any error conditions.

#### 3.5.2. Attack scenario.

According to Figure 9, the OCSP attack mode can be activated during the Null-Prefix attack specifically after step 6 when it is time to check if the certificate is still valid. The attack is based on intercepting OCSP response which for instance is intended for www.paypal.com server, and generating a forged one with response status as “tryLater”, which is simply the single ASCII character “3”. The user won’t notice anything and by activating this mode in sslniff tool, it will be very hard to revoke the null-prefix certificate, thus defeating OCSP [19].

#### 3.5.3. Evaluation.

Online certificate status protocol is vulnerable to this kind of attack because the “ResponseStatus” structure is not

<table>
<thead>
<tr>
<th>responseStatus</th>
<th>responseBytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCSPResponseStatus ::= ENUMERATED { successful (0), Response has valid confirmations malformedRequest (1), Illegal confirmation request internalError (2), Internal error in issuer tryLater (3), Try again later sigRequired (5), Must sign the request unauthorized (6), Request unauthorized }</td>
<td>BasicOCSPResponse ::= SEQUENCE { tbsResponseData ResponseData, signatureAlgorithm Algorithm Identifier, signature BIT STRING, cert [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL }</td>
</tr>
</tbody>
</table>

Table 1. Components of the OCSP response message.

Figure 10. The browsers that are vulnerable to null-prefix attack.

Figure 11. The browsers that are vulnerable to OCSP attack.
covered by a signature leaving it vulnerable to alteration by malicious users. Our study shows that the browsers in Figure 11 are vulnerable to this attack.

4. CONCLUSION

SSL has been long known for its Web transaction security applications. By deploying SSL into a web server, all the communication traffic between the client and server is secured, thus preventing attackers from tampering with data during transition. However, improper design of SSL in the various applications could lead to unexpected consequences. SSL sniffing, MD5 collide certificate, SSL stripping, SSL Null prefix, and OCSP attack are five examples of the most recent attacks exploiting design flaws of SSL. We implemented most of these attacks under Linux. We used different tools and procedures to successfully conduct these attacks and show their effectiveness. We also implemented our own parser to extract important data from the collected files. In case the password was encoded, we implemented a Java program to decode it. In short, the outcome of our implementation is the data transmitted between the client and server in clear text. More importantly, we evaluated the most popular browsers to know which of them is vulnerable to the discussed five attacks. The results show that all attacks except for SSL Sniffing can be performed on almost all browsers. This alarming fact is presented in details in Figure 12.

ACKNOWLEDGEMENTS

My special thanks go to Mouza Nayeli, Mariam Shamsi, and Eiman Sayari for their contribution to this work.

REFERENCES

5. Peter B. SSL Man-In-The-Middle Attacks, SANS Infosec Reading Room, February 2002.


