

Survey on SSL/TLS Attack Implementations

Vishal Menon [40220758], Ishan Chaudhari [40192935], Nithya Sri Bommakanti [40220572], Hima Bhatia [40226098], Himanshi Bahri [40219914], Syed Ahsan Abbas Naqvi [40168691], Sneha Mishra [40204368], Aayush Mehrotra [40191642]

ABSTRACT

SSL/TLS protocols are designed to keep communication as well as the transmission of data secure. These protocols and its implementations have become a necessary entity in every situation where communication exchange is involved. In this paper we have conducted a detailed study related to popular SSL/TLS protocols and implementations. This report reviews the details about each protocol mentioned within this report as well as vulnerabilities associated with the same that are known to exist. The vulnerabilities mentioned are gathered from both protocol and software based standpoints. The report also presents technical details associated with the flow of information and the various steps involved within the mentioned protocols.

I. INTRODUCTION

Secure communication over the internet is made possible by the widely used cryptographic technologies Transport Layer Security (TLS) and Secure Sockets Layer (SSL), which TLS replaced. They are made to make sure that information transferred between two parties is private and cannot be intercepted, altered, or read by unauthorized people. TLS/SSL has advantages for internet security, however, they are not impervious to attacks and flaws. Attackers have taken advantage of TLS/SSL flaws in some high-profile instances in recent years to steal sensitive data or launch cyberattacks. Man-in-the-middle (MitM) attacks are among the most frequent TLS/SSL attack types. An attacker can eavesdrop on a conversation between two people, manipulate the data being shared, or insert harmful and the most recent iteration, SSL3.0, was released by the company in 1996. The IETF organisation was responsible for the protocol's further development and release, but the protocol was subsequently renamed as TLS. Data in the transit layer is protected by this protocol. It offers security services for any application-based protocols, including HTTP, FTP, LDAP, POP3 and others. It is situated between the transport layer and the application layers in the ISO/OSI reference architecture. The protocol is used to establish a connection in a client/server environment and offers the following benefits to communication participants:

- Integrity
- Confidentiality
- Authentication

content into a transmission by intercepting it. Another attack is SSL stripping when the communication is downgraded from HTTPS to HTTP so that it is simpler to intercept and manipulate. This kind of attack is particularly harmful because it can go unnoticed by the user because the URL bar's padlock icon still makes the website appear secure. The POODLE attack, which uses a flaw in the SSLv3 protocol to decrypt sensitive information, and Heartbleed, which enables an attacker to extract sensitive information from the server's memory, are other sorts of attacks and vulnerabilities in existing libraries that can harm TLS/SSL. The goal of this paper is to give a general overview of some of the most prevalent TLS/SSL attack methods and vulnerabilities. We'll look at the technical specifics of these assaults and consider how they might affect the safety of web services and apps. We will also go through some preventative actions that may be done, such as using the most recent TLS/SSL versions and updating software regularly and security patches, to lessen the impact of these vulnerabilities. Ultimately, this paper will emphasize how crucial it is to comprehend the dangers posed by TLS/SSL and the precautions that may be done to maintain the security of sensitive data sent over the internet.

II. LITERATURE REVIEW

Background

A cryptographic protocol called SSL (Secure Sockets Layer), later renamed TLS (Transport Layer Security), is intended to guarantee the security of data transmitted over the Internet. This protocol was created by the Netscape business in 1994,

Methodology

To achieve the said objectives, the researchers adopted a simple yet effective methodology to develop and present a systematic review paper.

Literature search

As primary steps, this study searches the related literature using keywords from the database. The selected databases are Scopus and Google scholar, as these two databases are well known for their availability of a wide range of scientific articles, search are "SSL/TLS Protocols" and "vulnerabilities."

We carefully screen the title and abstract of articles to narrow down the quantity.

Literature analysis

After the screening stage, eight articles are carefully reviewed and analyzed to state the current state of the art in the literature on the Attacking SSL/TLS implementations.

Defining themes, approaches, and gaps

This study evaluates comparatively different researchers' different aims and objectives as well as unique research approaches. Therefore, this study reviews the selected papers about the shortlisted protocols and their vulnerabilities and aggregates the factors objectively.

Outline the structure

The results are then summarised and discussed within the group in order to show them in a more organised manner as a reviewed paper for our term project for the course INSE 6120.

The following paper is structured with eight sections, including an introduction where we introduce the SSL/TLS protocols and their background. Next, we describe all the attacks and vulnerabilities related to these attacks in a defined way. In the last section, we summarized our findings and concluded the topic.

III. STUDY ON VARIOUS SSL/TLS IMPLEMENTATIONS

A. OpenSSL

A comprehensive encryption tool called OpenSSL provides an open-source implementation for the TLS protocol. Users can create CSRs (Certificate Signing Requests), generate private keys, and activate SSL certificates, among other SSL-related duties.[6]

1. Uses

Demand for SSL credentials is currently strong. Since Google started its "HTTPS Everywhere" campaign, the security environment has undergone a significant shift.

To encourage the installation of digital certificates, they first improved SEO; later, Chrome effectively made HTTPS required for all users. Popular platforms like browsers such as Firefox and Chrome will flag your site as not secure if you don't use an SSL certificate.[4] With OpenSSL, you can set up the SSL files on your computer and register for a digital certificate (Generate the Certificate Signing Request). You can perform a variety of verifications and transform your certificate into different SSL forms.

Since not all sites offer user-friendly online interfaces for SSL administration, OpenSSL may be the only option for some platforms to import and set up your certificate.[6] The Open SSL instructions used on Linux systems are also used on Windows.

The TLS and SSL encryption algorithms are implemented in OpenSSL using open-source software. Basic cryptographic functions are implemented in the core package, which is written in the C computer language and offers several useful features. Several programming languages have wrappers that make it possible to use the OpenSSL library.

Applications that use OpenSSL are:

- HTTPS websites
- Internet Servers
- Email Servers
- VPN Software

2. Difference between SSL and OpenSSL

Secure Sockets Layer, also known as SSL, is a deprecated cryptographic system that protects network conversations between two computer apps [8]. When it comes to discuss SSL certificates, what we're referring to is TLS (Transport Layer Security) certificates since TLS is SSL's replacement. Alternatively, OpenSSL is a command-line tool for managing the creation, distribution, and authentication of SSL/TLS certificates.

OpenSSL version can be checked by the following command:

- `openssl version -a`

There are a few necessary stages to installing an SSL certificate on a website, and they apply to all servers and email clients. When someone does not have an online interface or wants to speed up the entire process, OpenSSL is particularly useful. To create the Certificate Signing Request as well as the secret key, combine files, one must check the details of the certificate, and resolve any possible problems OpenSSL commands.[8]

The CSR code can be generated using OpenSSL. A piece of text called a CSR contains information about the site and business. The Certification Authority must approve the CSR before someone can proceed. A person can also use an online CSR generator application to generate CSR. A secret key must be provided with the certificate request in order to generate the public key. It's advised to always produce a fresh private key at any time one wants to establish a CSR, even though they may utilize an existing key. It's time to make the CSR after the generation of the private key. It shall be in PEM form and contain information about the company in addition to the public key that was created using the private key provided.[11]

- `openssl req -new -key yourdomain.key -out yourdomain.csr`

One must select the key algorithm, key size, and possible password in order to create the private key. Although RSA is the common secret method, ECDSA

can also be used in some circumstances. When utilizing the RSA key technique, choose a key size of 2048 bits, whereas when employing the ECDSA key technique, choose a key size of 256 bits. Any key size below 2048 is not safe, and key sizes above 2048 may cause speed issues.

Finally, choose whether or not the secret key requires a passphrase. Some sites will not take passphrases and private keys.

- openssl genrsa -out yourdomain.key 2048

The data of the key can be viewed by the following:

- cat yourdomain.key

To make sure only the correct information is being submitted, we can verify the CSR information by the following command:

- openssl req -text -in yourdomain.csr -noout -verify

3. Major Vulnerabilities involving OpenSSL

a. Heartbleed

Through improper memory management in the TLS heartbeat extension, a flaw in OpenSSL could enable a remote attacker to reveal private information, potentially including user authentication passwords and secret keys. The TLS/DTLS pulse feature is implemented incorrectly in OpenSSL version 1.0.1. Using the vulnerable OpenSSL library, this vulnerability enables an attacker to obtain confidential memory of a program in segments of 64k at a time. You should be aware that a hacker can frequently exploit the flaw to recover as many 64k portions of memory as needed to obtain the desired secrets.[9] This flaw allows for the retrieval of the following private data:

- passwords and private keys used by susceptible sites
- Memory addresses and information used by susceptible services to access confidential data and get around attack mitigations.[7]

<p>alert tcp any [80,445] -> any [80,445] (msg:"FOX-SRT - Suspicious - Possible SSLv3 Large Heartbeat Response"; flow:established; ssl_version:ssl3; content:" 18 03 00 "; depth:3; byte_test2:>.200.3; byte_test2:<.17000.3; threshold:time limit, track by_src, count 1, seconds 600; reference:cve.2014-0160; classtype:bad-unknown; sid:21001128; rev:8)</p>
<p>alert tcp any [80,445] -> any [80,445] (msg:"FOX-SRT - Suspicious - Possible TLSv1 Large Heartbeat Response"; flow:established; ssl_version:tlsv1.0; content:" 18 03 01 "; depth:3; byte_test2:>.200.3; byte_test2:<.17000.3; threshold:time limit, track by_src, count 1, seconds 600; reference:cve.2014-0160; classtype:bad-unknown; sid:21001127; rev:8)</p>
<p>alert tcp any [80,445] -> any [80,445] (msg:"FOX-SRT - Suspicious - Possible TLSv1.1 Large Heartbeat Response"; flow:established; ssl_version:tlsv1.1; content:" 18 03 02 "; depth:3; byte_test2:>.200.3; byte_test2:<.17000.3; threshold:time limit, track by_src, count 1, seconds 600; reference:cve.2014-0160; classtype:bad-unknown; sid:21001128; rev:8)</p>
<p>alert tcp any [80,445] -> any [80,445] (msg:"FOX-SRT - Suspicious - Possible TLSv1.2 Large Heartbeat Response"; flow:established; ssl_version:tlsv1.2; content:" 18 03 03 "; depth:3; byte_test2:>.200.3; byte_test2:<.17000.3; threshold:time limit, track by_src, count 1, seconds 600; reference:cve.2014-0160; classtype:bad-unknown; sid:21001129; rev:8)</p>

Fig 2. Heartbleed Signatures

In order to establish a safe link using TLS, the exploit begins by transmitting a handshake to the website.

```
/* Read type and payload length first */
hbtype = *p++;
n2s(p, payload);
pl = p;

unsigned char *buffer, *bp;
int r;

/* Allocate memory for the response, size is 1 byte
 * message type, plus 2 bytes payload length, plus
 * payload, plus padding
 */
buffer = OPENSSL_malloc(1 + 2 + payload + padding);
bp = buffer;

/* Enter response type, length and copy payload */
*bp++ = TLS1_HB_RESPONSE;
s2n(payload, bp);
memcpy(bp, pl, payload);
```

Fig 1. Code snippet from Sean Cassidy's Heartbleed diagnosis.

If the person making the request wasn't truthful when she claimed to have provided the data bytes? What if pl only takes up one byte? Then, within the same operation, the read command from memcpy will read any memory that was close to the SSLv3 record. And it seems like there is a plethora of things close by.[10]

The actions listed below are the recommended measures for every SSL service that is at risk.

- OpenSSL should be updated to 1.0.1g
- Request the existing SSL certificate's cancellation.
- Make a new copy of your secret key.
- Ask for and change the SSL certificate

By looking at the network data, it is feasible to identify effective exploitation of this vulnerability. To identify successful abuse of the "heartbleed bug" researchers have created two groups of Snort signatures (Fig. 3). These can be used to detect any bleeds. The vulnerability is still active in a few webpages and can lead to issues from time to time. However, there are many defensive methods developed against this.[13]

b. CCS Injection Vulnerability

The well-known OpenSSL encryption software library has a severe flaw called the CCS Injection flaw. The SSL/TLS encryption system is implemented by OpenSSL and is used to safeguard the confidentiality of Internet contact. Numerous websites as well as programs like instant messaging, emails, and VPNs use OpenSSL. A skilled adversary can compel the implementation of weak keying material in OpenSSL SSL and TLS encryption clients as well as servers in some versions of the protocol. A man-in-the-middle

attack that allows the perpetrator to decrypt and alter data between the targeted client and server can take advantage of this.[4]

According to recent reports, this vulnerability has been around for at least ten years, but it was only recently discovered. Given the large number of companies that have deployed servers running OpenSSL, businesses that were concerned about or negatively impacted by Heartbleed bug might wish to take careful note about this susceptibility as well.[3] According to early accounts, attacks leave almost no evidence, making it very challenging, perhaps even unattainable, to identify their history. As a result, even if a company isn't susceptible right now, it might have been affected in the past. If this is the case, the company needs to take quick action to fix the problem. Only if the problem affects both the server and the client can it be abused. There is no danger of harm in a situation that just one or both of them is weak.[2] Masashi Kikuchi for identifying and studying this problem. On May 1, 2014, this problem was brought to OpenSSL's attention. Based on an initial patch from Masashi Kikuchi, Stephen Henson of the OpenSSL core team partially created the remedy. Only if the server is also prone to CVE-2014-0224 will services operating over SSL from PAN-OS / Panorama to third-party machines (such as syslog server, directory services server) be susceptible to a potential MITM attack. Verify that the machine hosting the third-party application is not using an outdated version of OpenSSL.[1]

```
-- Set a sufficiently large timeout
s:set_timeout(10000)

-- Send Client Hello to the target server
status, err = s:send(hello)
if not status then
    stdnse.debug1("Couldn't send Client Hello: %s", err)
    s:close()
    return false, Error.CONNECT
end

-- Read response
local done = false
local i = 1
local response
repeat
    status, response, err = tls.record_buffer(s, response, i)
    if err == "TIMEOUT" or not status then
        stdnse.verbose1("No response from server: %s", err)
        s:close()
        return false, Error.TIMEOUT
    end
    local record
    i, record = tls.record_read(response, i)
    if record == nil then
        stdnse.debug1("Unknown response from server")
        s:close()
        return false, Error.NOT_VULNERABLE
    elseif record.protocol == version then
        stdnse.debug1("Protocol version mismatch (%s)", version)
        s:close()
        return false, Error.PROTOCOL_MISMATCH
    end
    if record.type == "handshake" then
        for _, body in ipairs(record.body) do
            if body.type == "server_hello_done" then
                stdnse.debug1("Handshake completed (%s)", version)
                done = true
            end
        end
    end
end
```

Fig. 3. — Implementation of CCS Injection by Github User: r00t-3xp10it

c. *Drown*

DROWN or Decrypting RSA with Obsolete and Weakened Encryption corresponds to a cross-protocol security flaw that targets sites that support new SSLv3/TLS protocol packages by exploiting their support for the outdated, insecure SSL v2 protocol to launch assaults on connections that would normally be protected using more recent protocols. It makes it simpler for remote adversaries to decode TLS ciphertext data by using a Bleichenbacher RSA padding oracle because it requires a server to deliver a ServerVerify packet prior to verifying that an end user holds certain plaintext RSA data.[14]

When SSLv2 was in use, the master secret was immediately encrypted using RSA, and when 40-bit export ciphersuites were used, just 40 bits of the master secret were encrypted, leaving the remaining 88 bits unencrypted. The 48-byte SSLv3/TLS obfuscated RSA cipher is "trimmed" to 40-bit portions and is subsequently utilized for the SSLv2 ClientMasterKey message that the website's server considers as the 40-bit component of the SSLv2 master secret in which the rest of the 88 bits allow for anything provided by the user as unencrypted. It is possible to use the ServerVerify message to serve as oracle by employing brute force the 40-bit encryption.[12]

```
-- The length of clear_key is intentionally wrong to highlight the bug.
local clear_key = string.rep("\0", key_length - encrypted_key_length + 1)
local encrypted_key = string.rep("\0", encrypted_key_length)

local dummy_client_master_key = sslv2.client_master_secret(cipher, clear_key, encrypted_key)
socket:send(dummy_client_master_key)
local status, buffer, err = sslv2.record_buffer(socket, buffer, i)
socket:close()
if not status then
    return false
end
local i, message = sslv2.record_read(buffer, i)

-- Treat an error as a failure to force the cipher.
if not message or message.message_type == sslv2.SSL_MESSAGE_TYPES.ERROR then
    return false
end

return true
end
```

Fig. 4. — Implementation of DROWN vulnerability by Github User: bonsaiviking

- It makes it possible for man-in-the-middle attackers to bypass network security, eavesdrop, relay, and perhaps even change user and device messages.
- A potential attacker could access and steal confidential data.
- Attackers may pretend to be legitimate websites, steal their traffic, or modify their content.
- Data in ciphertext form could be decrypted by attackers.

Server administrators must make absolutely certain that their secret keys are never used with server-side software that supports SSLv2 communications in order to defend against DROWN. This applies to all applications that accept SSL/TLS, such as web servers, SMTP servers, IMAP and POP servers. In order to address the issue, the OpenSSL group has published a security warning and a number of updates that disable support for antiquated protocols and ciphers. However, the patched servers are still susceptible if the certificate of the server is utilized on other platforms that enable SSLv2. The vulnerability should be fixed as quickly as feasible by site administrators, according to numerous reports.[5]

B. LibreSSL

1. Overview

OpenSSL was affected by a severe vulnerability popularly known as the Heartbleed vulnerability. Due to the vulnerability mentioned and the multiple that followed affecting OpenSSL, multiple resolutions were applied under extreme urgency which led to an overall compromise of the security of OpenSSL by design. This is when LibreSSL was introduced as a fork of OpenSSL. It was intended and created to take the place of OpenSSL. As per the OpenBSD project, the goals of LibreSSL were to modernize the code, improve security and introduce best practices to software development. [15]

As far as modernization is concerned, LibreSSL attempts to counter the issues associated with support for C functions by using a compiler associated with the standard and also by assuming that the system that LibreSSL is going to be implemented on to be OpenBSD. In other words, due to conformity to the C standard, it allowed LibreSSL to get rid of code that was not associated with the standard. LibreSSL also removed the memory management layer that existed in OpenSSL which also helped discover multiple unreported buffer overflow errors which could then be resolved. One main goal of LibreSSL is to enhance security. This was achieved by improving the readability of the underlying code, getting rid of dangerous features and the cleaning of memory which is missing in OpenSSL. Unfortunately, even after the positive changes were introduced to combat the issues surrounding OpenSSL, LibreSSL was not able to become FIPS compliant simply because the FIPS object module was also one of the components that LibreSSL removed from the original OpenSSL implementation. [15]

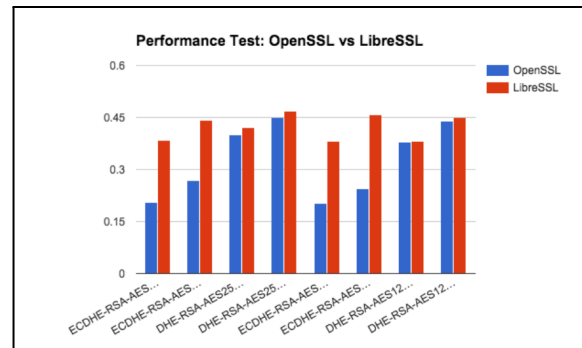


Fig 5: Performance test between OpenSSL and LibreSSL [15]

The above figure shows the performance difference between OpenSSL implementations compared against that of LibreSSL. The performance difference is based on the efficiency achieved by LibreSSL due to the memory sanitization within its design. Additionally, there are other features that have been heavily optimized for better performance as compared to OpenSSL.[15]

Applications that use LibreSSL:

- OpenBSD or other BSD based operating systems use this by default.
- Email Servers such as OpenSMTPD

2. Vulnerabilities

Among multiple vulnerabilities, two of the more popularly known vulnerabilities are mentioned below.

a. CVE 2014-9424

This is a vulnerability that causes a Denial-of-Service situation on the target machine. This is caused due to a double free vulnerability which is essentially a flaw that attempts to free a memory location twice. A function within versions of LibreSSL before the version 2.1.2 enables adversaries to DoS a system by making use of length-verification error which exists during the DTLS handshake. A DTLS handshake is essentially the preliminary connection request that is sent between the RDG server and the client to allow a secure encrypted connection.[60]

b. CVE 2015-5333:

The vulnerability marked by this CVE causes a buffer overflow vulnerability which ultimately leads to a DoS condition on the target machine. The vulnerability starts by calling the function OBJ_obj2txt() which was initially known to be vulnerable to a memory leak vulnerability as well. It is then accessed via the X509 function which is finally used to decode the X509 certificate that is present during the preliminary communication i.e. handshake.[61]

C. Microsoft Schannel

1. Overview

The Windows operating system includes a security package called Microsoft Schannel (Secure Channel) that supports secure communication protocols including SSL (Secure Sockets Layer) and TLS (Transport Layer Security). The negotiation and construction of secure connections between programs running on Windows-based PCs is managed by Schannel.[20]

Secure Internet communication is made possible by the cryptographic protocols and methods offered by Schannel. Moreover, it supports certificate-based authentication and network traffic encryption, shielding sensitive data from eavesdropping and manipulation.

Internet Explorer, Microsoft Edge, Microsoft Office, and Windows Server are just a few of the Microsoft products that use Schannel to connect securely to web servers, mail servers, and other network services. Developers can utilize Schannel's APIs to include secure communication in their own apps.

Several cryptographic techniques, including RSA, Diffie-Hellman, and elliptic curve encryption, are supported by the Schannel libraries (ECC). Digital signatures and certificate-based authentication are also supported by them.[20]

Microsoft updates the Schannel libraries frequently to fix security flaws, enhance performance, and increase interoperability with other programs and systems. To maintain the security and dependability of Windows-based systems, it is crucial to keep these libraries up to date. Some features of these libraries are:

- Implementation of the SSL/TLS protocols: The SSL (Secure Sockets Layer) and TLS (Transport Layer Security) protocols are implemented by the Microsoft Schannel libraries. Applications like web browsers, email clients, and instant messaging software all make use of these protocols to offer secure network connections.
- Security flaws: The POODLE and BEAST attacks, among others, have made Schannel libraries susceptible to a number of security flaws in the past. To fix these flaws and increase the security of the Schannel libraries, Microsoft publishes security updates and patches.
- Schannel libraries support a number of cryptographic protocols, including RSA, Diffie-Hellman, and elliptic curve cryptography (ECC). These methods enable secure key exchange and data encryption while communicating.
- Certificate-based authentication is a secure method of identifying a user or a system, and Schannel libraries support it. Digital files known as certificates are created by reputable organizations and contain data on the identity of

the user or machine. Throughout the authentication procedure, Schannel libraries check the validity of the certificates that the user or system presents.

- Schannel libraries include support for digital signatures, which are used to confirm the integrity and authenticity of digital data. A public key can be used to verify digital signatures, which are generated using a cryptographic method, and a private key.

In conclusion, Schannel is an important part of Windows' security infrastructure since it offers secure communication features necessary for safeguarding sensitive information and preserving the integrity of network traffic.

Applications that use Microsoft Schannel are:

- Now retired Internet Explorer and the new Edge browser
- Remote Desktop Protocols in windows.
- Windows update

2. Vulnerabilities

a. Beast attack:

The acronym BEAST stands for Browser Exploit Against SSL/TLS. It is a network vulnerability attack against TLS 1.0 and earlier SSL protocols. The attack was initially carried out in 2011 by security researchers Thai Duong and Juliano Rizzo, but Phillip Rogaway first identified the potential vulnerability in 2002. Man-in-the-middle attack approaches might allow attackers to listen in on the communication between a web server and a web browser. If they do and there is no encryption, they can see all the data sent back and forth between the web server and web browser, including passwords, credit card numbers, and other personal information even encryption, though, may have flaws and be broken.[16]

This is precisely what happened during the BEAST attack. The researchers discovered that TLS 1.0 (and earlier) encryption can be swiftly cracked, allowing the attacker to overhear the communication. A block cipher could be broken by the attacker by testing several combinations to determine if they provide the same outcome with the same initialization vector (which they know). Nevertheless, they can only do it for an entire block at once, and a block, for example, might include 16 bytes. As a result, the attacker would need to try 25616 combinations ($3.4028237e+38$) for each block in order to check the block.

The BEAST attack makes this considerably easier by requiring the attacker to guess a single byte at a time. This is possible if the attacker only requires one piece of sensitive information, such as a password, and can guess the majority of the data (such as HTML code, for example). The attacker can then carefully test

the encryption by choosing the proper data length such that they only have one byte of information in a block that they are unfamiliar with. The block can then be tested using only 256 different permutations of this byte. They then carry out the same procedure for the following byte, eventually obtaining the complete password.[19]

It is relatively simple to determine whether your web server is susceptible to BEAST. It is susceptible to BEAST if it supports TLS 1.0 or any SSL version. The RC4 cipher was initially advised as a defense against BEAST attacks (because it is a stream cipher, not a block cipher). RC4 was eventually discovered to be dangerous, though. The Payment Card Industry Data Security Standard (PCI DSS) currently forbids the usage of this encryption. As a result, you shouldn't ever defend yourself against BEAST using this technique.

Turning off TLS 1.0 and earlier protocols is the only straightforward cure for BEAST, just like it is for other network problems. For the most popular web server software, follow these steps. Moreover, we advise turning off TLS 1.1 and just enabling TLS 1.2. (all major browsers such as Google Chrome, Firefox, and Safari support TLS 1.2).

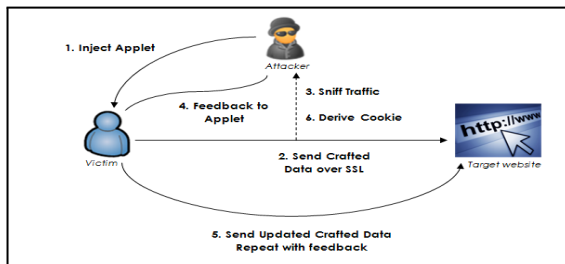


Fig 6: Information Flow in the BEAST attack

b. WinShock (CVE-2014-6321):

A security flaw called WinShock, often referred to as MS14-066, affects the Secure Channel (Schannel) security package in Microsoft Windows. The flaw was identified in 2014, and Microsoft gave it a critical rating. By delivering a carefully designed packet to the Schannel security package, the WinShock vulnerability enables an attacker to remotely execute code on a server running Microsoft IIS (Internet Information Services). As a result, the attacker may be able to take total control of the server, steal sensitive information, or introduce malware. All supported versions of Microsoft Windows, including Windows Server 2003, Windows Server 2008, Windows Server 2012, as well as Windows 7, 8, and 8.1, are impacted by the vulnerability. It results from a problem with the way Schannel manages packets that have been carefully constructed during the TLS/SSL handshake. In November 2014, Microsoft issued a security update to fix the WinShock flaw. Applying this update as soon as you can, will help stop attackers from taking advantage of the vulnerability. To further lower the

danger of exploitation, it is advised to adhere to best practices for secure application development, such as input validation and appropriate error handling.[17][18]

D. WolfSSL

1. Overview

WolfSSL is a lightweight SSL/TLS library written in C, primarily intended for use in embedded, IoT, and RTOS systems due to its minimal size, speed, portability, and feature set. It is an open-source implementation of TLS (SSL 3.0, TLS 1.0, 1.1, 1.2, and 1.3, and DTLS 1.0, 1.2, and 1.3) and supports various APIs, including those specified for SSL and TLS.[32] It also includes SSL/TLS client libraries and server implementations. The most popular OpenSSL features are included in the wolfSSL OpenSSL compatible interface.[21]

The idea for wolfSSL first came to Larry Stefonic and Todd Ouska in 2004, when they learned there was no other dual-licensed, open-source embedded SSL library on the market. At the time, OpenSSL was available, but many OpenSSL users desired a replacement that was readily portable, smaller, faster, accessible under a clear commercial license, furnished with a clean and modern API, and provided commercial-style developer support. [32]

WolfSSL was developed in response to this market demand and includes an OpenSSL compatibility layer. MySQL, the most widely used open-source database, was the first significant use of wolfSSL's SSL library. WolfSSL has attained astronomically high distribution numbers and user adoption through bundling with successful and well-liked open-source projects including MySQL, OpenWRT, Mongoose, cURL, and Ubuntu. WolfSSL presently protects more than 2 billion connections.[33]

Applications that use WolfSSL:

- Android NDK
- IoT Devices used in Smart Homes and Industrial IoT devices.
- VPN Software
- Online Gaming Servers

2. Protocol Vulnerabilities

a. CVE-2021-3674

This vulnerability, with a CVSS3 severity score of 9.8, affects wolfSSL's handling of DTLS handshake messages. The issue arises due to insufficient bounds checking on incoming handshake messages. An attacker could exploit this vulnerability by sending a malicious handshake message with a specially crafted size

field, causing a buffer overflow and potential remote code execution.

This vulnerability originated from an error in the code responsible for validating incoming DTLS handshake messages. Rizin was discovered to be flawed and by modifying the headers, the create section from the phdr function allots space for ELF section data. Out-of-bounds reads, which can result in memory corruption and possibly even code execution through the callback function of the binary object, can be brought on by intentionally constructed values in the headers. [22] Due to insufficient bounds checking, a specially crafted message could trigger a buffer overflow, leading to the execution of arbitrary code. An attacker could exploit this vulnerability by sending a malicious message to a vulnerable server or client, potentially compromising the system.[22]

b. CVE-2021-36234

This vulnerability concerns wolfSSL's processing of SSL/TLS session resumption requests. The problem emerges due to poor validation of session resumption tickets, which allows an attacker to create a ticket that may be used to impersonate a legitimate client or server.

This originated from a flaw in the code responsible for validating SSL/TLS session resumption tickets. The vulnerability is MIK.starlight 7.9.5.24363's usage of a hard-coded cryptographic key allows local users to decode credentials through unidentified vectors. [24]

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

Advisory ID:          SYSS-2021-039
Product:              MIK.starlight Server
Manufacturer:         MIK GmbH
Affected Version(s):  -
Tested Version(s):    7.9.5.24363
Vulnerability Type:    CWE-321: Use of Hard-Coded Cryptographic Key
Risk Level:           Medium
Solution Status:       Open
Manufacturer Notification: 2021-07-02
Solution Date:         -
Public Disclosure:     2021-08-27
CVE Reference:         CVE-2021-36234
Author of Advisory:    Nicola Staller, SySS GmbH
```

Fig 7: the signed PGP message providing details of the vulnerability [23]

During penetration testing, multiple flaws were identified on server MIK.starlight. Several operations on the MIK.starlight server are accessible via a WCF interface [25]. The "AddLogin" function is one of them. The function encrypts user credentials with a static encryption key before writing them to a file. So, whoever has access to this key can decode the credentials. One

more highly critical function was recognized among the sole functions designated for administrators. Administrators can read files from the file system by using "AdminGetFirstFileContentByFilePath" [25]. Arbitrary files can be read since the software is operated by a highly privileged user. This may, in addition to revealing sensitive information, enable remote code execution under certain conditions.[25]

3. Software Vulnerabilities

a. CVE-2020-24613

With a CVSS severity level of 9.8, this vulnerability impacts wolfSSL's TLS 1.3 implementation. The problem emerges as a result of incorrectly processing an erroneous signature in the certificate verify message, which might lead to remote code execution.

A major vulnerability was discovered in wolfSSL versions up to 4.4.x. This problem affects the method SanityCheckTls13MsgReceived in the file tls13.c of the TLS 1.3 Handler component. This vulnerability originated from a flaw in the code responsible for validating the signature in the certificate verification message during a TLS 1.3 handshake.[26] The use of an unknown input in the WAIT_CERT_CR parameter causes a certificate validation vulnerability. CWE-295 is the result of using CWE to declare the problem. A certificate is not validated or is validated improperly, by the program.[27]

The problem shown below is the method SanityCheckTls13MsgReceived() of file tls13.c.

```
case finished:
    #ifndef NO_WOLFSSL_CLIENT
    if (ssl->options.side == WOLFSSL_CLIENT_END) {
        if (ssl->options.clientState < CLIENT_HELLO_COMPLETE) {
            WOLFSSL_MSG("Finished received out of order");
            return OUT_OF_ORDER_E;
        }
        if (ssl->options.serverState <
            SERVER_ENCRYPTED_EXTENSIONS_COMPLETE) {
            WOLFSSL_MSG("Finished received out of order");
            return OUT_OF_ORDER_E;
        }
    }
    #endif
```

Fig 8: Vulnerable method SanityCheckTls13MsgReceived() [28]

b. CVE-2021-28157

An attacker could exploit this vulnerability by sending a specially crafted certificate verify message containing an invalid signature, leading

to a potential remote code execution on the vulnerable system.

This vulnerability affects wolfSSL's handling of RSA key exchange. The issue arises due to a flaw in the code of the devolutions server responsible for validating the RSA public key during a key exchange, leading to potential remote code execution. Devolutions Server has an extremely liberal CORS Policy that allows Cross-Origin requests from any domain. When a victim visits a malicious website, an attacker can interact with the API and exploit this vulnerability by submitting a specially generated RSA public key, potentially leading to remote code execution on the affected machine.[31]

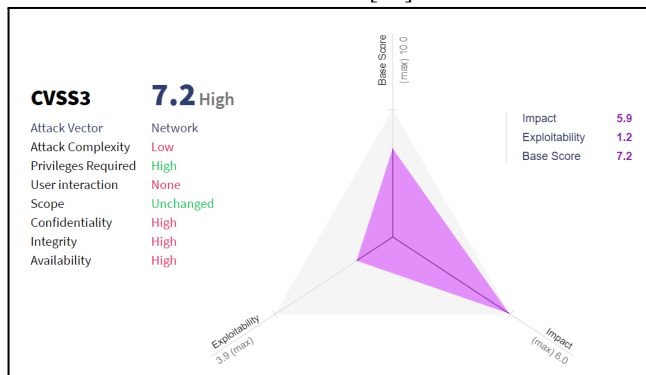


Fig 9: CVSS3 visual diagram of the impact and complexity [29]

An SQL Injection flaw in Devolutions Server before 2021.1 and Devolutions Server LTS prior to 2020.3.18 allows an administrator user to execute arbitrary SQL statements in `api/security/userinfo/delete` using a username [30]. With the user's deletion privileges, an administrator has the ability to insert SQL code into a username and run it. [31] This could allow an attacker to execute arbitrary code using a vulnerable system, potentially leading to data theft, system compromise, or other malicious activity.

E. GNUTLS

1. Overview

GnuTLS is a secure communications library that implements and supports the SSL, TLS, and DTLS protocols and technologies. It includes a simple C language application programming interface (API) for accessing secure communication protocols, as well as APIs for parsing and writing X. 509, PKCS #12, and other necessary structures. GnuTLS was designed to provide security against attacks like eavesdropping, tampering and damaging the message integrity. It provides secure communication for the backend and works with all base linux libraries.[34]

GnuTLS is made up of 3 independent parts, TLS protocol part, Certificate part and cryptographic backend part. The "TLS protocol portion" is the actual protocol implementation,

which is completed entirely within the GnuTLS library. Verification functionality and certificate parsing is provided by the certificate part, which gets its functionality from `libtasn1` library. Data confidentiality and integrity is provided by the cryptographic part which obtains its functionality from `nettle` and `gmplib` libraries. Following are some of the important features of GnuTLS library:

- Support for TLS 1.3, TLS 1.2, TLS 1.1, TLS 1.0 and optionally SSL 3.0 protocols.
- Support for Datagram TLS 1.0 and 1.2.
- Support for handling and verification of X.509 certificates.
- Support for password authentication using TLS-SRP

GnuTLS follows a development cycle in which even minor version numbers represent a stable release and odd minor version numbers represent a development release. `Nettle` and `gmplib` are the base libraries for installation of GnuTLS, before installing GnuTLS installation of libraries is necessary.

Following are some of the useful commands:

- `Enable/disable-srp-authentication`
- `Enable/disable-psk-authentication`
- `Enable/disable-anon-authentication`
- `Enable/disable-dhe`
- `Enable/disable-ecdh`
- `Enable/disable-openssl-compatibility`
- `Enable/disable-dtls-srtp-support`
- `Enable/disable-alpn-support`
- `Enable/disable-heartbeat-support`

Applications of GNUTLS:

- Web Browsers such as Chromium and its derivatives.
- Instant messaging services such as Empathy.
- Remote Access tools that work with Libreswan(VPN) and OpenVPN.
- Package managers like `apt-get` and `pacman` in Debian and Arch Linux.

2. Functionality of GNUTLS

There is a global state for read-only that starts once the global initialization function starts. This global structure contains memory allocation functions, ASN.1 parser structures, and, depending on the system's CPU, references to hardware accelerated encryption functions. There is no major change in global GnuTLS function except one change and that is the decentralization function which must be called once all the programs are permanently furnished to free the allocated memory.[35]

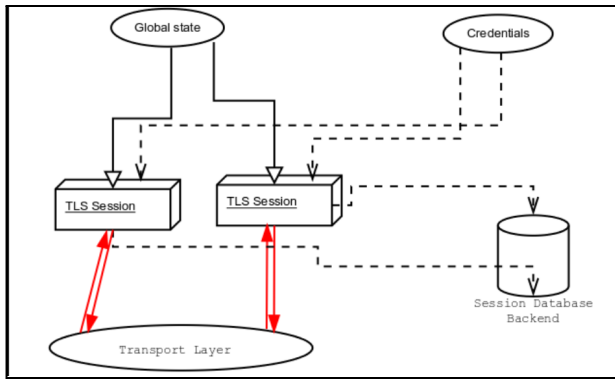


Fig 9: Information Flow in GNUTLS

Authentication methods like certificate authentication are verified through credentials structure. This structure stores the parameters like private keys, certificates and other information that is used to prove the identity of the user.

Session state and all other relevant information about the sessions are stored in the session database. Each session has a unique session ID which is used for session identification purposes. After successful session handshake, each GnuTLS session calls the backend function to perform the appropriate functionality. The server examines the session database immediately after receiving the client hello, and if the session ID supplied by the client matches a stored session, the saved session is retrieved, and the new session is a resumed one with the same session ID as the previous one.[36]

By design, the GnuTLS library is thread safe, which means that library objects, such as TLS sessions, can be safely partitioned among threads as long as a single thread accesses a single object. This is enough to support a server that supports many sessions per thread. Thread-safe read-only access to objects, such as credentials holding structures, is also possible.[37]

Error code	Error	Description
0	GNUTLS_E_SUCCESS	Success
-3	GNUTLS_E_UNKNOWN_COMPRESSION_ALGORITHM	Unable to negotiate the supported compression method
-6	GNUTLS_E_UNKNOWN_CIPHER_TYPE	Incompatible cipher type
-7	GNUTLS_E_LARGE_PACKET	Large packet size
-8	GNUTLS_E_UNSUPPORTED_VERSION_PACKET	Packet with illegal or unsupported version was received
-9	GNUTLS_E_UNEXPECTED_PACKET_LENGTH	Error decoding received TLS packet
-10	GNUTLS_E_INVALID_SESSION	The specified session has been invalidated for some reason

Fig 10: Error codes and its descriptions

3. Vulnerabilities

a. CVE-2022-2509 - Double Free

Operating system allocates some memory to store data temporarily and free it after some time and the same memory block is allocated by the operating system to the programmer for temporary storage purposes. Ideally, the memory allocation software will detect that the block no longer belongs to the section of the program that is "returning" it, will recognize that the block has already been recycled, and will refuse to deallocate it a second time,

avoiding the hazards of "freeing" it again. But if the memory allocator is not properly handled and it hands out the same memory spot to another program at exactly the same loaned out spot then this causes great problems. Attackers can take advantage of this bug and you could end up with the same memory chunk exploited with the two parts of the same program. This vulnerability allows an attacker to run malicious code on the target system. This bug existed in certificate verification code. The solution to mitigate this vulnerability is to upgrade to latest version of GnuTLS

b. CVE-2020-11501 - Incorrect Cryptographic Function

Before GnuTLS version 3.6.13, version used incorrect cryptographic function in DTLS. Instead of using random value DTLS clients were using 32 '\0' bytes value, this made it easier for attackers to brute force as there was no randomness in DTLS negotiation. If the attacker gets successful in getting this number they can significantly damage the system files and corrupt the system. This vulnerability could be mitigated by updating the GnuTLS version to any version after 3.6.13.[33]

F. MatrixSSL

1. Overview

MatrixSSL is the first open-source small footprint SSL stack and is also known as Inside Secure TLS Toolkit. It is offered in the form of a Software Development Kit (SDK) that is geared towards application in Internet of Things (IoT) devices and other embedded systems. An open-source TLS/SSL implementation is made for use in embedded hardware settings. For Internet of Things (IoT) devices with small footprints and minimal connection overhead, MatrixSSL is an embedded SSL and TLS implementation with minimum dependencies on portability.

MatrixSSL uses public key algorithms such as RSA, Diffie-Hellman Algorithm, etc. and symmetric algorithms such as AES, ChaCha20-Poly1305 and other algorithms as well as client and server support for TLS 1.3, mutual authentication, session resumption, and other features. There are portability levels for additional OS systems, cypher suites, and cryptography providers in the well-documented source, which also includes these features.

2. Vulnerabilities

a. CVE-2022-43974

It is a buffer overflow vulnerability found in MatrixSSL versions 4.5.1-4.0.0 that could allow information disclosure and remote code execution. An attacker could use a network link to overwrite the data in RAM on a server running MatrixSSL if

there was a buffer overflow (TLS Toolkit). This vulnerability has been demonstrated to be usable for a denial-of-service attack. Additionally, it might be possible for an attacker to exploit this vulnerability to install and execute malicious code. It is feasible to trick the TLS1.3 'change cypher spec' processing to result in an integer overflow by using a specifically crafted packet.[44] All MatrixSSL (TLS Toolkit) versions that enable TLS1.3 have the matrixSslDecodeTls13() function implemented incorrectly. It was found and reported by security analysts Robert Hörr and Alissar Ibrahim from Deutsche Telekom's IT Security Evaluation Facility, and was fixed in version 4.6.0, which was made available in December 2022.

b. CVE-2019-14431

The DTLS server faces issues while handling inbound network messages in MatrixSSL versions 3.8.3 Open through 4.2.1 Open and hence causing a heap-based buffer overflow of up to 256 bytes and Remote Code Execution. During this process of mishandling of the crafted data packets, the server handles the fragment length value provided with the DTLS messages in an incorrect manner.[46]

```
Error message WITHOUT Address Sanitizer:

matrixssl-4-2-1-open: apps/dtls/dtlsServer -p 4444
DTLS server running on port 4444
Select user 1
Got REQUEST_RECV from ReceivedData
*** Error in 'apps/dtls/dtlsServer': malloc(): Memory corruption: 0x00000000142d9e8 ***
***** BackTrace: *****
/lib/x86_64-linux-gnu/libc.so.6(+0x77f45)[0x7fcaee41745]
/lib/x86_64-linux-gnu/libc.so.6(+0x0213e)[0x7fcaee4c13e]
/lib/x86_64-linux-gnu/libc.so.6(__libc_calculate_tmbuf)[0x7fcaee4e0c0]
apps/dtls/dtlsServer[0x411405]
apps/dtls/dtlsServer[0x416115]
apps/dtls/dtlsServer[0x416144]
apps/dtls/dtlsServer[0x4020c6]
/lib/x86_64-linux-gnu/libc.so.6(__libc_start_main+0xf9)[0x7fcaee4a830]
apps/dtls/dtlsServer[0x402095]
***** Memory Map: *****
00400000-004a3000 r-xp 00000000 fd:01 273412      matrixssl-4-2-1-open/apps/dtls/dtlsServer
004a3000-004a3000 r-p 000a2000 fd:01 273412      matrixssl-4-2-1-open/apps/dtls/dtlsServer
004a3000-004a4000 rw-p 000a3000 fd:01 273412      matrixssl-4-2-1-open/apps/dtls/dtlsServer
004a4000-004a5000 rw-p 00000000 00:00 0
004a5000-01440000 rw-p 00000000 00:00 0
7fcaee0000-7fcaee01000 r-xp 00000000 00:00 0      [heap]
7fcaee01000-7fcaee02000 --p 00000000 00:00 0
7fcaee02000-7fcaee2a000 r-xp 00000000 fd:01 2039      /lib/x86_64-linux-gnu/libgcc_s.so.1
7fcaee2a000-7fcaee2b000 --p 00010000 fd:01 2039      /lib/x86_64-linux-gnu/libgcc_s.so.1
7fcaee2b000-7fcaee2c000 r-xp 00010000 fd:01 2039      /lib/x86_64-linux-gnu/libgcc_s.so.1
7fcaee2c000-7fcaee2d000 r-xp 00000000 fd:01 2037      /lib/x86_64-linux-gnu/libc-2.23.so
7fcaee2d000-7fcaee2e000 --p 00010000 fd:01 2037      /lib/x86_64-linux-gnu/libc-2.23.so
7fcaee2e000-7fcaee2f000 r-xp 00010000 fd:01 2037      /lib/x86_64-linux-gnu/libc-2.23.so
7fcaee2f000-7fcaee30000 r-xp 00010000 fd:01 2037      /lib/x86_64-linux-gnu/libc-2.23.so
7fcaee30000-7fcaee31000 r-xp 00000000 00:00 0
7fcaee31000-7fcaee32000 r-xp 00000000 fd:01 2037      /lib/x86_64-linux-gnu/libpthread-2.23.so
7fcaee32000-7fcaee33000 r-xp 00000000 fd:01 2037      /lib/x86_64-linux-gnu/libpthread-2.23.so
7fcaee33000-7fcaee34000 r-xp 00000000 fd:01 2037      /lib/x86_64-linux-gnu/libpthread-2.23.so
7fcaee34000-7fcaee35000 r-xp 00000000 fd:01 2037      /lib/x86_64-linux-gnu/libpthread-2.23.so
7fcaee35000-7fcaee36000 r-xp 00000000 fd:01 2037      /lib/x86_64-linux-gnu/libpthread-2.23.so
7fcaee36000-7fcaee37000 r-xp 00000000 fd:01 2037      /lib/x86_64-linux-gnu/libpthread-2.23.so
7fcaee37000-7fcaee38000 r-xp 00000000 fd:01 2037      /lib/x86_64-linux-gnu/libpthread-2.23.so
7fcaee38000-7fcaee39000 r-xp 00000000 00:00 0
7fcaee39000-7fcaee3a000 r-xp 00000000 00:00 0
7fcaee3a000-7fcaee3b000 r-xp 00000000 00:00 0
7fcaee3b000-7fcaee3c000 r-xp 00000000 00:00 0
7fcaee3c000-7fcaee3d000 r-xp 00000000 00:00 0
7fcaee3d000-7fcaee3e000 r-xp 00000000 00:00 0
7fcaee3e000-7fcaee3f000 r-xp 00000000 00:00 0
7fcaee3f000-7fcaee40000 r-xp 00000000 00:00 0
7fcaee40000-7fcaee41000 r-xp 00000000 00:00 0
7fcaee41000-7fcaee42000 r-xp 00000000 00:00 0
7fcaee42000-7fcaee43000 r-xp 00000000 00:00 0
7fcaee43000-7fcaee44000 r-xp 00000000 00:00 0
7fcaee44000-7fcaee45000 r-xp 00000000 00:00 0
7fcaee45000-7fcaee46000 r-xp 00000000 00:00 0
7fcaee46000-7fcaee47000 r-xp 00000000 00:00 0
7fcaee47000-7fcaee48000 r-xp 00000000 00:00 0
7fcaee48000-7fcaee49000 r-xp 00000000 00:00 0
7fcaee49000-7fcaee4a000 r-xp 00000000 00:00 0
7fcaee4a000-7fcaee4b000 r-xp 00000000 00:00 0
7fcaee4b000-7fcaee4c000 r-xp 00000000 00:00 0
7fcaee4c000-7fcaee4d000 r-xp 00000000 00:00 0
7fcaee4d000-7fcaee4e000 r-xp 00000000 00:00 0
7fcaee4e000-7fcaee4f000 r-xp 00000000 00:00 0
7fcaee4f000-7fcaee50000 r-xp 00000000 00:00 0
7fcaee50000-7fcaee51000 r-xp 00000000 00:00 0
7fcaee51000-7fcaee52000 r-xp 00000000 00:00 0
7fcaee52000-7fcaee53000 r-xp 00000000 00:00 0
7fcaee53000-7fcaee54000 r-xp 00000000 00:00 0
7fcaee54000-7fcaee55000 r-xp 00000000 00:00 0
7fcaee55000-7fcaee56000 r-xp 00000000 00:00 0
7fcaee56000-7fcaee57000 r-xp 00000000 00:00 0
7fcaee57000-7fcaee58000 r-xp 00000000 00:00 0
7fcaee58000-7fcaee59000 r-xp 00000000 00:00 0
7fcaee59000-7fcaee5a000 r-xp 00000000 00:00 0
7fcaee5a000-7fcaee5b000 r-xp 00000000 00:00 0
7fcaee5b000-7fcaee5c000 r-xp 00000000 00:00 0
7fcaee5c000-7fcaee5d000 r-xp 00000000 00:00 0
7fcaee5d000-7fcaee5e000 r-xp 00000000 00:00 0
7fcaee5e000-7fcaee5f000 r-xp 00000000 00:00 0
7fcaee5f000-7fcaee60000 r-xp 00000000 00:00 0
7fcaee60000-7fcaee61000 r-xp 00000000 00:00 0
7fcaee61000-7fcaee62000 r-xp 00000000 00:00 0
7fcaee62000-7fcaee63000 r-xp 00000000 00:00 0
7fcaee63000-7fcaee64000 r-xp 00000000 00:00 0
7fcaee64000-7fcaee65000 r-xp 00000000 00:00 0
7fcaee65000-7fcaee66000 r-xp 00000000 00:00 0
7fcaee66000-7fcaee67000 r-xp 00000000 00:00 0
7fcaee67000-7fcaee68000 r-xp 00000000 00:00 0
7fcaee68000-7fcaee69000 r-xp 00000000 00:00 0
7fcaee69000-7fcaee6a000 r-xp 00000000 00:00 0
7fcaee6a000-7fcaee6b000 r-xp 00000000 00:00 0
7fcaee6b000-7fcaee6c000 r-xp 00000000 00:00 0
7fcaee6c000-7fcaee6d000 r-xp 00000000 00:00 0
7fcaee6d000-7fcaee6e000 r-xp 00000000 00:00 0
7fcaee6e000-7fcaee6f000 r-xp 00000000 00:00 0
7fcaee6f000-7fcaee70000 r-xp 00000000 00:00 0
7fcaee70000-7fcaee71000 r-xp 00000000 00:00 0
7fcaee71000-7fcaee72000 r-xp 00000000 00:00 0
7fcaee72000-7fcaee73000 r-xp 00000000 00:00 0
7fcaee73000-7fcaee74000 r-xp 00000000 00:00 0
7fcaee74000-7fcaee75000 r-xp 00000000 00:00 0
7fcaee75000-7fcaee76000 r-xp 00000000 00:00 0
7fcaee76000-7fcaee77000 r-xp 00000000 00:00 0
7fcaee77000-7fcaee78000 r-xp 00000000 00:00 0
7fcaee78000-7fcaee79000 r-xp 00000000 00:00 0
7fcaee79000-7fcaee7a000 r-xp 00000000 00:00 0
7fcaee7a000-7fcaee7b000 r-xp 00000000 00:00 0
7fcaee7b000-7fcaee7c000 r-xp 00000000 00:00 0
7fcaee7c000-7fcaee7d000 r-xp 00000000 00:00 0
7fcaee7d000-7fcaee7e000 r-xp 00000000 00:00 0
7fcaee7e000-7fcaee7f000 r-xp 00000000 00:00 0
7fcaee7f000-7fcaee80000 r-xp 00000000 00:00 0
7fcaee80000-7fcaee81000 r-xp 00000000 00:00 0
7fcaee81000-7fcaee82000 r-xp 00000000 00:00 0
7fcaee82000-7fcaee83000 r-xp 00000000 00:00 0
7fcaee83000-7fcaee84000 r-xp 00000000 00:00 0
7fcaee84000-7fcaee85000 r-xp 00000000 00:00 0
7fcaee85000-7fcaee86000 r-xp 00000000 00:00 0
7fcaee86000-7fcaee87000 r-xp 00000000 00:00 0
7fcaee87000-7fcaee88000 r-xp 00000000 00:00 0
7fcaee88000-7fcaee89000 r-xp 00000000 00:00 0
7fcaee89000-7fcaee8a000 r-xp 00000000 00:00 0
7fcaee8a000-7fcaee8b000 r-xp 00000000 00:00 0
7fcaee8b000-7fcaee8c000 r-xp 00000000 00:00 0
7fcaee8c000-7fcaee8d000 r-xp 00000000 00:00 0
7fcaee8d000-7fcaee8e000 r-xp 00000000 00:00 0
7fcaee8e000-7fcaee8f000 r-xp 00000000 00:00 0
7fcaee8f000-7fcaee90000 r-xp 00000000 00:00 0
7fcaee90000-7fcaee91000 r-xp 00000000 00:00 0
7fcaee91000-7fcaee92000 r-xp 00000000 00:00 0
7fcaee92000-7fcaee93000 r-xp 00000000 00:00 0
7fcaee93000-7fcaee94000 r-xp 00000000 00:00 0
7fcaee94000-7fcaee95000 r-xp 00000000 00:00 0
7fcaee95000-7fcaee96000 r-xp 00000000 00:00 0
7fcaee96000-7fcaee97000 r-xp 00000000 00:00 0
7fcaee97000-7fcaee98000 r-xp 00000000 00:00 0
7fcaee98000-7fcaee99000 r-xp 00000000 00:00 0
7fcaee99000-7fcaee9a000 r-xp 00000000 00:00 0
7fcaee9a000-7fcaee9b000 r-xp 00000000 00:00 0
7fcaee9b000-7fcaee9c000 r-xp 00000000 00:00 0
7fcaee9c000-7fcaee9d000 r-xp 00000000 00:00 0
7fcaee9d000-7fcaee9e000 r-xp 00000000 00:00 0
7fcaee9e000-7fcaee9f000 r-xp 00000000 00:00 0
7fcaee9f000-7fcaeea0000 r-xp 00000000 00:00 0
7fcaeea0000-7fcaeea1000 r-xp 00000000 00:00 0
7fcaeea1000-7fcaeea2000 r-xp 00000000 00:00 0
7fcaeea2000-7fcaeea3000 r-xp 00000000 00:00 0
7fcaeea3000-7fcaeea4000 r-xp 00000000 00:00 0
7fcaeea4000-7fcaeea5000 r-xp 00000000 00:00 0
7fcaeea5000-7fcaeea6000 r-xp 00000000 00:00 0
7fcaeea6000-7fcaeea7000 r-xp 00000000 00:00 0
7fcaeea7000-7fcaeea8000 r-xp 00000000 00:00 0
7fcaeea8000-7fcaeea9000 r-xp 00000000 00:00 0
7fcaeea9000-7fcaeeaa000 r-xp 00000000 00:00 0
7fcaeeaa000-7fcaeeab000 r-xp 00000000 00:00 0
7fcaeeab000-7fcaeeac000 r-xp 00000000 00:00 0
7fcaeeac000-7fcaeead000 r-xp 00000000 00:00 0
7fcaeead000-7fcaeeae000 r-xp 00000000 00:00 0
7fcaeeae000-7fcaeeaf000 r-xp 00000000 00:00 0
7fcaeeaf000-7fcaeeb0000 r-xp 00000000 00:00 0
7fcaeeb0000-7fcaeeb1000 r-xp 00000000 00:00 0
7fcaeeb1000-7fcaeeb2000 r-xp 00000000 00:00 0
7fcaeeb2000-7fcaeeb3000 r-xp 00000000 00:00 0
7fcaeeb3000-7fcaeeb4000 r-xp 00000000 00:00 0
7fcaeeb4000-7fcaeeb5000 r-xp 00000000 00:00 0
7fcaeeb5000-7fcaeeb6000 r-xp 00000000 00:00 0
7fcaeeb6000-7fcaeeb7000 r-xp 00000000 00:00 0
7fcaeeb7000-7fcaeeb8000 r-xp 00000000 00:00 0
7fcaeeb8000-7fcaeeb9000 r-xp 00000000 00:00 0
7fcaeeb9000-7fcaeeba000 r-xp 00000000 00:00 0
7fcaeeba000-7fcaeebb000 r-xp 00000000 00:00 0
7fcaeebb000-7fcaeebc000 r-xp 00000000 00:00 0
7fcaeebc000-7fcaeebd000 r-xp 00000000 00:00 0
7fcaeebd000-7fcaeebe000 r-xp 00000000 00:00 0
7fcaeebe000-7fcaeebf000 r-xp 00000000 00:00 0
7fcaeebf000-7fcaeec0000 r-xp 00000000 00:00 0
7fcaeec0000-7fcaeec1000 r-xp 00000000 00:00 0
7fcaeec1000-7fcaeec2000 r-xp 00000000 00:00 0
7fcaeec2000-7fcaeec3000 r-xp 00000000 00:00 0
7fcaeec3000-7fcaeec4000 r-xp 00000000 00:00 0
7fcaeec4000-7fcaeec5000 r-xp 00000000 00:00 0
7fcaeec5000-7fcaeec6000 r-xp 00000000 00:00 0
7fcaeec6000-7fcaeec7000 r-xp 00000000 00:00 0
7fcaeec7000-7fcaeec8000 r-xp 00000000 00:00 0
7fcaeec8000-7fcaeec9000 r-xp 00000000 00:00 0
7fcaeec9000-7fcaeeca000 r-xp 00000000 00:00 0
7fcaeeca000-7fcaeecb000 r-xp 00000000 00:00 0
7fcaeecb000-7fcaeecc000 r-xp 00000000 00:00 0
7fcaeecc000-7fcaeece000 r-xp 00000000 00:00 0
7fcaeece000-7fcaeecf000 r-xp 00000000 00:00 0
7fcaeece000-7fcaeed0000 r-xp 00000000 00:00 0
7fcaeed0000-7fcaeed1000 r-xp 00000000 00:00 0
7fcaeed1000-7fcaeed2000 r-xp 00000000 00:00 0
7fcaeed2000-7fcaeed3000 r-xp 00000000 00:00 0
7fcaeed3000-7fcaeed4000 r-xp 00000000 00:00 0
7fcaeed4000-7fcaeed5000 r-xp 00000000 00:00 0
7fcaeed5000-7fcaeed6000 r-xp 00000000 00:00 0
7fcaeed6000-7fcaeed7000 r-xp 00000000 00:00 0
7fcaeed7000-7fcaeed8000 r-xp 00000000 00:00 0
7fcaeed8000-7fcaeed9000 r-xp 00000000 00:00 0
7fcaeed9000-7fcaeeda000 r-xp 00000000 00:00 0
7fcaeeda000-7fcaeedb000 r-xp 00000000 00:00 0
7fcaeedb000-7fcaeedc000 r-xp 00000000 00:00 0
7fcaeedc000-7fcaeedd000 r-xp 00000000 00:00 0
7fcaeedd000-7fcaeede000 r-xp 00000000 00:00 0
7fcaeede000-7fcaeedf000 r-xp 00000000 00:00 0
7fcaeedf000-7fcaeeef000 r-xp 00000000 00:00 0
7fcaeeef000-7fcaeeef000 r-xp 00000000 00:00 0
Aborted
```

Fig 11: Error message in MatrixSSL without using address sanitizer

```
Error message WITH Address Sanitizer:

--17575==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x60200000d11 at pc 0x7f28a227994 bp 0x7f6d1495f68 s
WRITE of size 256 at 0x60200000d11 thread 0
#0 0x7f28a227993 in __asan_memcpy (/usr/lib/x86_64-linux-gnu/libasan.so.2+0x8c9b3)
#1 0x4010b0 in memcpy (/usr/include/x86_64-linux-gnu/libc.so.2+0x1b3)
#2 0x4010b0 in parseSSLHandshake matrixssl-4-2-1-open/matrixssl/sslDecode.c:2400
#3 0x4010b0 in matrixsslDecodeTls13AndBelow matrixssl-4-2-1-open/matrixssl/sslDecode.c:1433
#4 0x402007 in matrixsslReceiveData matrixssl-4-2-1-open/matrixssl/matrixssl.c:1301
#5 0x406007 in main matrixssl-4-2-1-open/apps/dtls/dtlsServer.c:899
#6 0x7f28a1a6082f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.2+0x282f)
#7 0x40a9f8 in _start (matrixssl-4-2-1-open/test_matrixssl_asan.exe+0x40a9f8)

0x60200000d11 is located 0 bytes to the right of 1-byte region [0x60200000d10,0x60200000d11)
allocated by thread 0 here:
#0 0x7f28a22e3802 in malloc (/usr/lib/x86_64-linux-gnu/libasan.so.2+0x98602)
#1 0x404001 in parseSSLHandshake matrixssl-4-2-1-open/matrixssl/sslDecode.c:2344
#2 0x404001 in matrixsslDecodeTls13AndBelow matrixssl-4-2-1-open/matrixssl/sslDecode.c:1433

SUMMARY: AddressSanitizer: heap-buffer-overflow 77:0 __asan_memcpy
Shadow bytes around the buggy address:
0x0c047fff9b30: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c047fff9b40: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c047fff9b50: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c047fff9b60: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c047fff9b70: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c047fff9b80: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c047fff9b90: fa fa 00 00 fa fa 00 00 fa fa 00 00 fa fa 00 00
0x0c047fff9ba0: fa fa 00 00 fa fa 00 00 fa fa 00 00 fa fa 00 00
0x0c047fff9bb0: fa fa 00 00 fa fa 00 00 fa fa 00 00 fa fa 00 00
0x0c047fff9bc0: fa fa 00 00 fa fa 00 00 fa fa 00 00 fa fa 00 00
0x0c047fff9bd0: fa fa 00 00 fa fa 00 00 fa fa 00 00 fa fa 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
Addressable: 00
Partially addressable: 01 02 03 04 05 06 07
Heap left redzone: fa
Heap right redzone: fb
Freed heap redzone: fd
Stack left redzone: f1
Stack mid redzone: f2
Stack right redzone: f3
Stack partial redzone: f4
Stack after return: f5
Stack use after scope: f8
Global redzone: f9
Global init order: fe
Poisoned by user: f7
Container overflow: fc
Array cookie: ac
Intra object redzone: bb
ASAN internal: fe
--17575==ABORTING
```

Fig 12: Error message in MatrixSSL after using address sanitizer

c. CVE-2018-12439

This vulnerability uses the Return of The Hidden Number Problem, also known as ROHNP to challenge ECDSA signatures using MatrixSSL through version 3.9.5 Open. The attacker requires access to either the local computer or another virtual machine running on the same physical host in order to find out an ECDSA key.[45] The vulnerability exposes sensitive information to an attacker or the outside world that is not explicitly authorized to have access to that information.

d. CVE-2017-1000415

The SSL/TLS protocol has long utilised the X.509 Public-Key Infrastructure to accomplish authentication. Its popularity in recent times has been further boosted by a new trend called Internet-of-Things (IoT) devices using small footprint SSL/TLS libraries for secure communication. The X.509 security assurances rely on the underlying implementation's strict examination of X.509 certificate chains and acceptance of only legitimate ones. Implementations of X.509 that are not compliant run the risk of causing attacks and/or compatibility problems.[62]

This vulnerability is related to the X.509 certificate validation process in MatrixSSL. The X.509 certificate validation procedure in MatrixSSL version 3.7.2 incorrectly validates UTCTime date ranges, which causes some certificates to have their expiration (beginning) year extended (delayed) by 100 years.

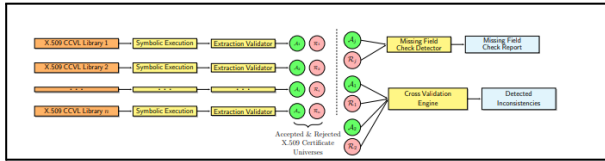


Fig 13: Noncompliance finding approach for X.509 CCVL implementations.

G. Apple Secure Transport

1. Overview

Secure network communication using standardized transport layer security mechanisms. The Security.SecureTransport API gives you access to Apple's implementation of Secure Sockets Layer version 3.0 (SSLv3), Transport Layer Security (TLS) versions 1.0 through 1.2, and Datagram Transport Layer Security (DTLS) version 1.0.

Below are some instances of protocol-based and software-based vulnerabilities in the Apple SecureTransport library for the SSL/TLS protocol.[47]

Applications that use Apple Secure Transport are:

- Safari Web Browsers
- Facetime
- iMessage

2. Vulnerabilities

a. "FREAK" vulnerability (CVE-2015-0204)

The SSL/TLS protocol had a design issue that allowed attackers to coerce clients into using poor encryption ciphers, which could then be used to intercept and decrypt SSL/TLS traffic.

Researchers revealed the FREAK attack, a new SSL/TLS vulnerability, on March 3, 2015, on Tuesday. As a result, an attacker can force susceptible clients and servers to utilize weakened encryption, which the attacker can then crack to steal or alter sensitive data. This enables an attacker to intercept HTTPS communications between vulnerable clients and servers.[48]

The term "FREAK vulnerability" describes a flaw in the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols that is brought on using "export-grade" encryption. "Factoring RSA Export Keys" is the name's acronym.

The threat enables a hacker to force a weak client to employ a weaker key exchange cipher, which would let them to intercept data transmission. Because the RSA encryption can be easily cracked when utilising 512 bits or fewer.

In general, the FREAK vulnerability makes it possible for hackers to intercept HTTPS traffic between clients and susceptible servers to access the private key of a website. As a result, they are now able to decode login cookies, passwords, credit card

numbers, and other sensitive data from HTTPS connections.

The client is compelled to utilise an "export-grade" key, often known as a 512-bit export RSA key, which is significantly simpler to trace and crack than current encryption standards, thereby compromising secure connections.

In place of the conventional RSA cipher suites, an attacker can request "export RSA" via the client's Welcome message. The server then responds with a 512-bit export cipher key rather than the high-security keys used today. The answer is signed using its permanent key.

The Man-in-the-Middle attacker can obtain the RSA decryption key and utilise the "pre-master secret" to gain access to the TLS' master secret, which is used for symmetric encryption of messages in the connection, because the website client accepts the weak "export-grade" key. The core of command injection threats is when an attacker inserts harmful code into a plaintext file.

Millions of users have been exposed for years due to the FREAK vulnerability. Due to the vulnerability's size, it affected the entire sector and had a high potential for damaging cyberattacks. For instance, OpenSSL is used by numerous programmes, including browsers for Android. Apple's Secure transport, on the other hand, is used in both iOS and OS X applications, affecting iPhones, iPads, and Macs.

The FREAK issue was soon addressed by updates from OpenSSL, Google, and Apple.

The problem with computers and mobile devices was resolved by updates released by Apple's product security team. Additionally, it was discovered that the Safari browser proved resistant to the danger.

It was urged to turn off support for any ciphers whose security was in doubt, including export versions of cypher suites. Also, there was detailed configuration advice for default configurations and suggestions for cypher suite enforcement laws.[49]

b. "Logjam" vulnerability (CVE-2015-4000)

A design weakness in the SSL/TLS protocol led to this vulnerability because it permitted attackers to utilise a weaker encryption protocol, which could then be used to intercept and decrypt SSL/TLS traffic.

c. "POODLE" vulnerability (CVE-2014-3566)

Similarly, because of the design defect in the SSL/TLS protocol, attackers were able to intercept and decode SSL/TLS traffic by taking advantage of a hole in the SSLv3 protocol.

d. "Gotofail" vulnerability (CVE-2014-1266)

A programming issue in the SecureTransport library led to SSL/TLS connections being formed without sufficient certificate checking, leading to the "Gotofail" vulnerability.



```
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRes, SSLBuffer signedParams,
                                uint8_t *signature, UInt16 signatureLen)
{
    OSStatus      err;
    SSLBuffer      hashOut, hashCtx, clientRandom, serverRandom;
    uint8_t        hashes[SSL_SHA1_DIGEST_LEN + SSL_MD5_DIGEST_LEN];
    SSLBuffer      signedHashes;
    uint8_t        dataToSign;
    size_t         dataToSignLen;

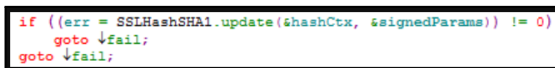
    --
    if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signature)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;

    err = sslRawVerify(ctx,
                      ctx->peerPubKey,
                      dataToSign,           /* plaintext */
                      dataToSignLen,       /* plaintext length */
                      signature,
                      signatureLen);

    if (err) {
        sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
                    "returned %d\n", (int)err);
        goto fail;
    }
}
```

Fig 14: It shows there was a bug in the implementation of SSLVerifySignedSeverKeyExchange function.

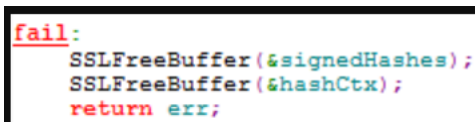
The issue was the two consecutive goto fail; statements. The second line is run regardless of whether the predicate in the if-statement is true or false, despite the indentation of the lines making it appear as though they will both be executed only if the predicate is true. Correcting the indentation makes the issue more visible.



```
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
goto fail;
```

Fig 15: Error in code with two consecutive goto fail statements.

When we call SSLHashSHA1.update it will return an error, the value of err will almost always be zero when the second goto fails; the statement is executed. But when goto fail is executed, the caller gets a return value as zero, who believes the signature verification of the "Server Key Exchange" passed.



```
fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
```

Fig 16: Execution of Goto Fail statement

This vulnerability enables man-in-the-middle attack, the attackers spoof SSL servers by –

- By using an arbitrary private key for the signing step or
- Omitting the signing step.

The problem could be prevented in several ways such as –

- Writing/compiling code.
- Reviewing code
- Testing

To overcome this vulnerability end users should apply the patches provided by Apple.[50][51]

e. "Triple Handshake" vulnerability

A programming mistake in the SecureTransport library led to the establishment of SSL/TLS connections using a less secure handshake protocol, which may be used to intercept and decrypt SSL/TLS communication.

f. "Heartbleed" vulnerability (CVE-2014-0160)

As previously explained under Section A "Vulnerabilities involving OpenSSL", the SecureTransport library for OS X, which makes use of the OpenSSL library, contained a bug that led to this vulnerability. By taking advantage of a weakness in the way OpenSSL processed specific heartbeat messages, the attackers were able to retrieve critical information from memory.

To fix these flaws in their devices, Apple has released patches and updates. In all the above vulnerabilities except Heartbleed both iOS and OS X devices were impacted.

H. Mozilla NSS

1. Overview

Mozilla NSS (Network Security Services) is a set of libraries that provide cryptographic and security functionality for applications. The NSS library provides several cryptographic features such as SSL/TLS, Cryptographic algorithms, Certificate management and Key management, thereby, making it powerful and flexible for developers to implement strong cryptographic security [52]. NSS can be used on various operating systems such as Windows, Linux, and macOS and is licensed under Additionally, NSS is licensed under the permissive open-source license i.e., Mozilla Public License [53].

Applications that use NSS are:

- Mozilla products such as Firefox, SeaMonkey.
- AOL Communicator & AOL Instant Messenger
- Open source products like Pidgin & Evolution.

2. Vulnerabilities

a. CVE-2021-43527

It is a software-based vulnerability that affects server as well as client applications using NSS versions before 3.73 ESR. The way NSS verifies certificates has been determined to have a remote code execution issue [54]. When a client application built using NSS tries to establish an SSL/TLS connection, this weakness enables an attacker acting

as an SSL/TLS server to cause this problem. The weakness can also be exploited by a server application built using NSS that processes client certificates and receives a rogue certificate from a client. Confidentiality, integrity, and system availability pose the most risk to this vulnerability. [55] [56]

When handling DSA or RSA-PSS signatures that have been DER-encoded, NSS (Network Security Services) versions are susceptible to a heap overflow. Applications that use NSS for signatures encoded in CMS, S/MIME, PKCS #7, or PKCS #12 are likely to be impacted. Additionally, applications using NSS for certificate validation or other TLS, X.509, OCSP or CRL functionality are affected. This basically depends on how NSS is installed.[58]

Red Hat, Thunderbird, LibreOffice, Evolution, and Evince are among the programmes that are vulnerable since they use NSS for signature verification, in contrast to Mozilla's Firefox web browser, email client, and PDF viewers, which are secure from the vulnerability.[58]

The vulnerability is due to a bug in the certificate verification code in the vfy_CreateContext function of the secvfy.c file. The error manifests itself both when the client reads the certificate from the server as well as when the server processes the client's certificates.[57]

In order to prevent this vulnerability, first, we should verify if Mozilla NSS is being used in organisations applications using SSL/TLS. Then, the latest security patches should be applied after proper testing. In order to mitigate the attack, all the software should be run as a common user. Last, least privilege should be granted to all the accounts.

b. Protocol Based Vulnerability

The SSLv2 protocol's design includes a flaw since client master keys, whose authenticity is decided by the first step of the SSL handshake, are used to produce session keys. Buffer overflow is brought on by the client master key's incorrect length. An exploitable buffer overflow exists.

Using the privileges of the user executing the application, the attacker can run the code. Moreover, a hacker might launch a denial-of-service attack.

By turning off SSLv2 support, this vulnerability can be protected against. Firefox 2.0.0.2, Firefox 1.5.0.10, SeaMonkey 1.0.8, and NSS 3.11.5 all get around this.[59]

IV. CONCLUSION

Secure online communication is now more important than ever in the current digital era. It is crucial to make sure that data carried over the internet is secure to prevent snoopers from accessing it because of the rise in e-commerce, internet

banking, and other confidential online transactions. SSL/TLS offers safe internet connection by shielding data from interception, modification, and eavesdropping. However, SSL/TLS has encountered numerous attacks and flaws over the years, despite its usefulness. These flaws have brought to light the significance of secure implementation and frequent updates. For this project, we reviewed a few of the noteworthy vulnerabilities because of which attackers were able to obtain confidential data, reduce encryption levels, and sometimes even decipher SSL/TLS connections. In order to make sure that security flaws are fixed, and that powerful cryptographic techniques and protocols are employed, SSL/TLS installations must be periodically updated.

SSL/TLS technology is constantly being innovated upon and looks to have a bright future ahead. These days, more secure encryption protocols and algorithms like ChaCha20 and TLS 1.3 are being developed. These new innovations provide enhanced security, quicker performance, and better defense against new threats. Moreover, SSL/TLS implementations are updated more regularly to fix known flaws and stop the emergence of brand-new ones. Despite these developments, SSL/TLS technology still has room for improvement. The usage of insecure encryption techniques and protocols, which attackers can take advantage of, is one of the key causes for concern. Also, website owners and developers need to be better informed on the value of SSL/TLS security as well as how to apply it correctly. These problems can be solved, ensuring that SSL/TLS is a dependable as well as reliable technology for many years to come.

In summary, SSL/TLS technology has had a number of weaknesses over the years, with ongoing attempts to bolster and increase its security, the technology's future appears bright. We can make sure that SSL/TLS is a dependable and trustworthy system for the future by resolving its flaws and putting best practices into practice.

REFERENCES

- [1] S. (n.d.). *GitHub - secretnonempty/CVE-2014-0224*. GitHub. <https://github.com/secretnonempty/CVE-2014-0224>
- [2] *CVE-2014-0224 OpenSSL Man-in-the-middle vulnerability*. (2014, June 9). CVE-2014-0224 OpenSSL Man-in-the-middle Vulnerability. <https://security.paloaltonetworks.com/CVE-2014-0224>
- [3] *OpenSSL CCS Injection Vulnerability (CVE-2014-0224) Alert - Red Hat Customer Portal*. (2016, May 20). Red Hat Customer Portal. <https://access.redhat.com/articles/904433>
- [4] *DCX*. (n.d.). DCX. https://success.trendmicro.com/dcx/s/solution/1103813-trend-micro-products-and-the-ccs-injection-vulnerability--cve-2014-0224-openssl-vulnerability?language=en_US&sfidcIframeOrigin=null#:~:text=What%20is%20the%20CCS%20Injection,the%20privacy%20of%20Internet%20communication
- [5] Langley, A. (n.d.). *ImperialViolet - Early ChangeCipherSpec Attack*. ImperialViolet - Early ChangeCipherSpec Attack. <https://www.imperialviolet.org/2014/06/05/earlyccs.html>
- [6] Inc., O. F. (n.d.). */index.html*. /index.html. <https://www.openssl.org/>
- [7] Inc., O. F. (n.d.). */news/vulnerabilities.html*. /News/vulnerabilities.html. <https://www.openssl.org/news/vulnerabilities.html>
- [8] *OpenSSL - Wikipedia*. (1998, January 1). OpenSSL - Wikipedia. <https://en.wikipedia.org/wiki/OpenSSL>

- [9] Fruhlinger, J. (n.d.). *The Heartbleed bug: How a flaw in OpenSSL caused a security crisis*. CSO Online. <https://www.csoonline.com/article/3223203/the-heartbleed-bug-how-a-flaw-in-openssl-caused-a-security-crisis.html>
- [10] *The Heartbleed Bug, explained*. (2014, June 19). Vox. <https://www.vox.com/2014/6/19/18076318/heartbleed>
- [11] *OpenSSL "Heartbleed" vulnerability (CVE-2014-0160) | CISA*. (2016, October 5). Cybersecurity and Infrastructure Security Agency CISA. <https://www.cisa.gov/news-events/alerts/2014/04/08/openssl-heartbleed-vulnerability-cve-2014-0160>
- [12] R. X. (n.d.). *RC-exploiter/ssl-ccs-injection.nse at master · r00t-3xp10it/RC-exploiter*. GitHub. <https://github.com/r00t-3xp10it/RC-exploiter>
- [13] *sean cassidy : Diagnosis of the OpenSSL Heartbleed Bug*. (2014, April 7). Sean Cassidy : Diagnosis of the OpenSSL Heartbleed Bug. <https://www.seancassidy.me/diagnosis-of-the-openssl-heartbleed-bug.html>
- [14] *SSLv2-Drown Vulnerability in OpenSSL | Trend Micro Help Center*. (n.d.). SSLv2-Drown Vulnerability in OpenSSL | Trend Micro Help Center. <https://helpcenter.trendmicro.com/en-us/article/tmka-19804>
- [15] *LibreSSL: The Secure OpenSSL Alternative | Infosec Resources*. (2023, January 23). Infosec Resources. <https://resources.infosecinstitute.com/topic/libressl-the-secure-openssl-alternative/>
- [16] Balasino, A. (2013, December 2). *SSL/TLS attacks: Part 1 – BEAST Attack - Checkmate*. Checkmate. <https://niiconsulting.com/checkmate/2013/12/ssl-tls-attacks-part-1-beast-attack/>
- [17] *WinShock: What is it? How to Remediate CVE-2014-6321*. (n.d.). Rapid7. <https://www.rapid7.com/resources/winshock-what-is-it-how-to-remediate/>
- [18] *SSL/TLS security: Addressing WinShock, the Schannel vulnerability | TechTarget*. (2015, February 1). Security. <https://www.techtarget.com/searchsecurity/tip/SSL-TLS-security-Addressing-WinShock-the-Schannel-vulnerability>
- [19] Nidecki, T. A. (2020, May 21). *What Is the BEAST Attack | Acunetix*. Acunetix. <https://www.acunetix.com/blog/web-security-zone/what-is-beast-attack/>
- [20] A. P. (2023, February 14). *TLS/SSL overview (Schannel SSP)*. TLS/SSL Overview (Schannel SSP) | Microsoft Learn. <https://learn.microsoft.com/en-us/windows-server/security/tls/tls-ssl-schannel-ssp-overview>
- [21] Drake, Nate. "What Is WolfSSL?" *TechRadar*, 3 Mar. 2023, www.techradar.com/vpn/what-is-wolfssl
- [22] "NVD - CVE-2021-3674." *Nvd.nist.gov*, nvd.nist.gov/vuln/detail/CVE-2021-3674
- [23] *SySS.de*, 2021, www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2021-039.txt
- [24] "NVD - CVE-2021-36234." *Nvd.nist.gov*, nvd.nist.gov/vuln/detail/CVE-2021-36234
- [25] Staller, Nicola. "Multiple Vulnerabilities in MIK.starlight Server (SYSS-2021-035, SYSS-2021-036, SYSS-2021-037, SYSS-2021-038, SYSS-2021-039)." *SySS Tech Blog*, 2 Oct. 2021, blog.syss.com/posts/syss-2021-035-039/
- [26] "NVD - CVE-2020-24613." *Nvd.nist.gov*, nvd.nist.gov/vuln/detail/CVE-2020-24613. Accessed 2 Apr. 20
- [27] "CVE-2020-24613 | WolfSSL TLS 1.3 Tls13.c SanityCheckTls13MsgReceived Certificate Validation." *Vuldb.com*, vuldb.com/?id.160214
- [28] "Technical Advisory – WolfSSL TLS 1.3 Client Man-In-The-Middle Attack (CVE-2020-24613)." *NCC Group Research Blog*, 24 Aug. 2020, research.nccgroup.com/2020/08/24/technical-advisory-wolfssl-tls-1-3-client-man-in-the-middle-attack/
- [29] "Debricked." *Debricked.com*, debricked.com/vulnerability-database/vulnerability/CVE-2021-28157
- [30] "NVD - CVE-2021-28157." *Nvd.nist.gov*, nvd.nist.gov/vuln/detail/CVE-2021-28157
- [31] "DEVO-2021-0004." *Devolutions*, devolutions.net/security/advisories/DEVO-2021-0004
- [32] "WolfSSL." *Wikipedia*, 3 Jan. 2023, en.wikipedia.org/wiki/WolfSSL
- [33] *WolfSSL – Embedded SSL/TLS Library*. www.wolfssl.com/
- [34] *GnuTLS 3.8.0*. (n.d.). GnuTLS 3.8.0. <https://gnutls.org/manual/gnutls.html#Introduction-to-GnuTLS>
- [35] *GnuTLS 3.8.0*. (n.d.). GnuTLS 3.8.0. <https://gnutls.org/manual/gnutls.html#How-to-use-GnuTLS-in-applications>
- [36] Ducklin, P. (2022, August 1). *GnuTLS patches memory mismanagement bug – update now!* Naked Security. <https://nakedsecurity.sophos.com/2022/08/01/gnutls-patches-memory-mismanagement-bug-update-now/>
- [37] *GnuTLS*. (n.d.). GnuTLS. <https://gnutls.org/security-new.html>
- [38] *CVE-2020-11501 : GnuTLS 3.6.x before 3.6.13 uses incorrect cryptography for DTLS. The earliest affected version is 3.6.3 (2018-07-16) bec*. (n.d.). CVE-2020-11501 : GnuTLS 3.6.x Before 3.6.13 Uses Incorrect Cryptography for DTLS. the Earliest Affected Version Is 3.6.3 (2018-07-16) Bec. <https://www.cvedetails.com/cve/CVE-2020-11501/>
- [39] *Debian -- Security Information -- DSA-4652-1 gnutls28*. (n.d.). Debian -- Security Information -- DSA-4652-1 Gnutls28. <https://www.debian.org/security/2020/dsa-4652>
- [40] *Red Hat Customer Portal - Access to 24x7 support and knowledge*. (n.d.). Red Hat Customer Portal - Access to 24x7 Support and Knowledge. <https://access.redhat.com/security/cve/cve-2019-3836>
- [41] <https://www.nccgroup.trust/us/our-research/technical-advisory-return-of-the-hidden-number-problem/>. (n.d.). <https://www.nccgroup.trust/us/our-research/technical-advisory-return-of-the-hidden-number-problem/>
- [42] Zorz, Z. (2023, January 13). *Vulnerabilities in cryptographic libraries found through modern fuzzing - Help Net Security*. Help Net Security. <https://www.helpnetsecurity.com/2023/01/13/fuzzing-cryptographic-libraries/>
- [43] *NVD - CVE-2022-43974*. (n.d.). NVD - CVE-2022-43974. <https://nvd.nist.gov/vuln/detail/CVE-2022-43974>
- [44] M. (2023, January 2). *Buffer Overflow in MatrixSSL*. GitHub. <https://github.com/matrixssl/matrixssl/security/advisories/GHSA-fmwc-gwc5-2g29>
- [45] C., & C. (n.d.). *CVE-2018-12439*. CVE.report. <https://cve.report/CVE-2018-12439>
- [46] *NVD - CVE-2019-14431*. (n.d.). NVD - CVE-2019-14431. <https://nvd.nist.gov/vuln/detail/CVE-2019-14431>
- [47] *Secure Transport | Apple Developer Documentation*. (n.d.). Apple Developer Documentation. https://docs.developer.apple.com/documentation/security/secure_transport
- [48] *Tracking the FREAK Attack*. (n.d.). Tracking the FREAK Attack. <https://freakattack.com/>
- [49] *FREAK Vulnerability - What it is and how to prevent it*. (2021, April 2). Crashtest Security. <https://crashtest-security.com/prevent-ssl-freak/>
- [50] Team, S. E., Bals, F., Hogg, M., Research Center, S. C., & Freeman, C. (2014, February 25). *Understanding the Apple 'goto fail;' vulnerability*. Application Security Blog. <https://www.synopsys.com/blogs/software-security/understanding-apple-goto-fail-vulnerability-2/>
- [51] *NVD - CVE-2014-1266*. (n.d.). NVD - CVE-2014-1266. <https://nvd.nist.gov/vuln/detail/CVE-2014-1266>
- [52] *Network Security Services (NSS)*. (n.d.). Network Security Services (NSS). <https://www-archive.mozilla.org/projects/security/pki/nss/>
- [53] *Network Security Services (NSS) & Firefox Source Docs documentation*. (n.d.). Network Security Services (NSS) & Firefox Source Docs Documentation. <https://firefox-source-docs.mozilla.org/security/nss/index.html>
- [54] Kumar, S. (2021, December 6). *Vulnerability in Mozilla's NSS Crypto Library impacts software*. The Cybersecurity Daily News. <https://cyberdaily.securelayer7.net/vulnerability-in-mozilla-nss-crypto-library-impacts-software/>
- [55] *Red Hat Customer Portal - Access to 24x7 support and knowledge*. (n.d.). Red Hat Customer Portal - Access to 24x7 Support and Knowledge. <https://access.redhat.com/security/cve/cve-2021-43527>

- [56] *RHSB-2021-008 NSS Memory corruption when decoding DSA signatures (CVE-2021-43527) - Red Hat Customer Portal.* (2022, February 15). Red Hat Customer Portal.
<https://access.redhat.com/security/vulnerabilities/RHSB-2021-008>
- [57] R. (2021, December 1). *This shouldn't have happened: A vulnerability postmortem.* Project Zero: This Shouldn't Have Happened: A Vulnerability Postmortem.
<https://googleprojectzero.blogspot.com/2021/12/this-shouldnt-have-happened.html>
- [58] D. (2021, December 1). *BigSig, a vulnerability in Mozilla NSS that could allow code execution.* Linux Adictos.
<https://www.linuxadictos.com/en/bigsig-a-vulnerability-in-mozilla-nss-that-could-allow-code-execution.html>
- [59] *CERT/CC Vulnerability Note VU#592796.* (2007, March 7). VU#592796 - Mozilla Network Security Services (NSS) Fails to Properly Handle the Client Master Key. <https://www.kb.cert.org>
- [60] "Red Hat Customer Portal - Access to 24x7 support and knowledge," access.redhat.com, Dec. 28, 2014.
<https://access.redhat.com/security/cve/CVE-2014-9424> (accessed Mar. 19, 2023).
- [61] "Full Disclosure: Qualys Security Advisory - LibreSSL (CVE-2015-5333 and CVE-2015-5334)," seclists.org, Oct. 15, 2015.
<https://seclists.org/fulldisclosure/2015/Oct/75> (accessed Mar. 19, 2023).
- [62] S. Y. Chau et al., "SymCerts: Practical Symbolic Execution for Exposing Noncompliance in X.509 Certificate Validation Implementations," *IEEE Xplore*, May 01, 2017.
<https://ieeexplore.ieee.org/document/7958595> (accessed Apr. 04, 2023).