null – The Open Security Community

Learning from the Past:

Cryptography Vulnerabilities Exploited in 2021

White Paper

Contents

1. About null

null - The open security community is one of the most active and vibrant communities of cybersecurity professionals. Started with the simple idea of providing a knowledge-sharing platform to cybersecurity professionals, null has grown many folds. Currently, null has 20+ active chapters and organizes many security events for aspiring cybersecurity professionals. null is about spreading information security awareness. All our activities such as null Monthly Meets, null Humla, null Bachaav, null Puliya, null Job Portal are for the cause of that.

null is now focusing on contributing to enterprise security. Working on many projects to collaborate with the enterprise, such as:

- Defining security frameworks and standards for producing security guidelines in the upcoming IT technology like Cloud, Blockchain/Cryptography, IoT, AI/ML, and many more.
- Develop tools and methodologies in order to secure the products and infrastructure based on the above-mentioned technologies.
- Start many new security projects and publish research papers.

This white paper is one small effort to our contribution to achieving the above objectives.

2. Acknowledgement

On behalf of **null - The open security community**, we would like to thank the authors of this white paper, who have contributed their precious time and effort to publish this paper.

Authors

- **Bhavesh Dhake** (https://www.linkedin.com/in/bdhake/)
- **Praneeth Varma** (https://www.linkedin.com/in/srikrishpraneeth/)
- **Sagar Yadwad** (https://www.linkedin.com/in/sagar-yadwad-79711123)
- **Naman Agrawal** (https://www.linkedin.com/in/naman-agrawal1778)
- **Ajit Hatti** (https://www.linkedin.com/in/ajithatti/)

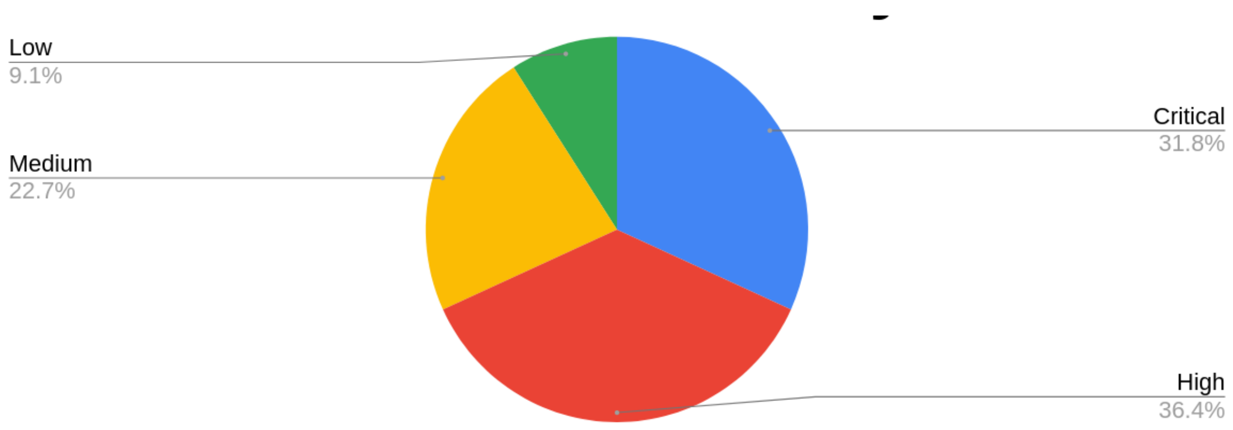
Project Co-ordinator

- **Murtuja Bharmal** (https://www.linkedin.com/in/murtujabharmal/)

3. Abstract

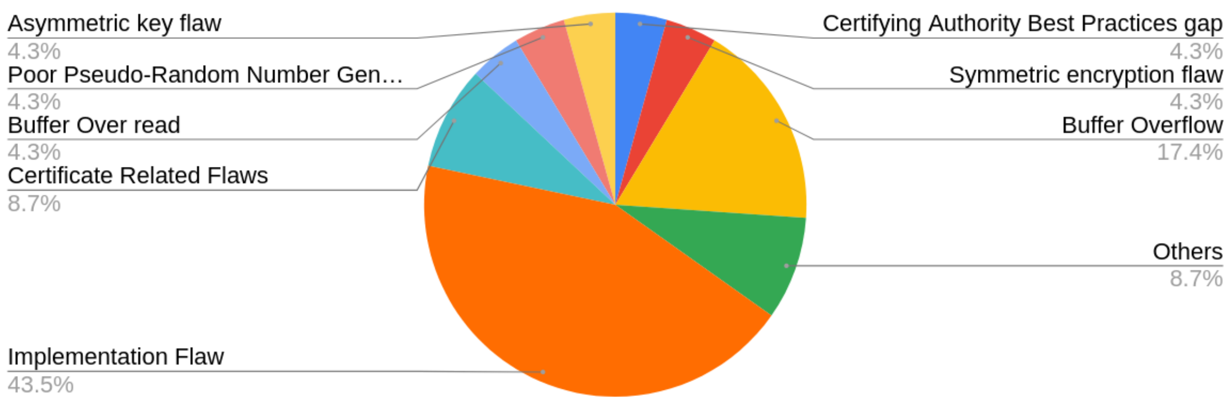
This is the third edition of the Null Community’s Annual Cryptography Security Report. The previous reports for the year 2019 & 2020 are very good resources to learn about the incidents, innovative attacks and concepts around secure implementation of cryptography.

For the year 2021, we tracked 25 security incidents of significance with severity ranging from High (36%) to Critical (32%). These incidents also resulted in major operational disruptions and downtimes at affected organizations.



Severity Distribution of Cryptography related Incidents in 2021

We learned that most of the vulnerabilities were implementation gaps (43%) & coding flaws (17%). These are the flaws which can be avoided as well as easily detected with the better awareness for Best Practices for Secure Cryptographic Implementation.



Cryptography related incidents, distribution based on category 2021

Certificate Authorities (CA) are the backbone of the world's Security & Trust Architecture. Since we started the NCSC initiative in 2019, we have been seeing a significant number of incidents where CAs were found deviating from the best practices recommended by the **Certification Authority Browser Forum** (CA/B Forum). Such incidents have very high security, trust and financial impact (check Verisign/Symantec story).

We have been seeing a significant reduction in CA best practices related incidents, year (12% in 2020) on year (4% in 2021). This is really good news for the community and the entire cyber world.

With that good news, we aim to deliver even better contributions from the Null community. We are working together to improve awareness, build knowledge base, tools and expertise to improve the security of cryptographic Implementations and this report is one part of such efforts.

This report will help the audience to understand the root cause of the incidents better and gain from the research and hard work put together by the **Null Cipher Security Club** (NCSC).

We welcome your participation, suggestions and criticism, reach us at void@null.co.in. With this report in your hand, we wish you happy learning and better cipher security ahead.

Cheers,

Null Cipher Security Club

4. Motivation

Cryptography being a foundational discipline in cybersecurity, the discussion around its vulnerabilities, good practices standardization in practices is left to the cryptographer who are not the practitioners. Practitioners of cybersecurity are focused on all conventional vulnerabilities except for cryptographic vulnerabilities.

This paper is an attempt to address the gap and initiate discussion & interest of the security community at wider realm to adopt and practise the affinity for cryptography & its secure implementation.

Leveraging the forum of NULL open security community, we attempt to analyse the cryptographic vulnerabilities of 2021 which have been impacting the year 2022 and we expect them to continue to do the same until we seriously pay attention to better practices and procedure when it comes to cryptography implementation and securing the cryptographic resources.

5. Overview

We have collected and organized most of the cryptography implementation related vulnerabilities which are exploited in 2021. We have classified these vulnerabilities into below mentioned major categories :

1. Implementational flaws
2. Certificate Related Flaws
3. Buffer Overflow
4. Certifying Authority Best Practices gap
5. Poor Pseudo-Random number generation
6. Asymmetric key flaw
7. Symmetric encryption flaw
8. Others

For all the vulnerabilities we have provided the following details :

1. Vulnerability description
2. Severity
3. How it can be exploited
4. Impact
5. How to mitigate it

A brief about each vulnerability can be found in the below section.

6. Vulnerability Details

6.1 Implementation flaw

6.1.1 CVE-2021-23839 Incorrect SSLv2 rollback protection

Introduction

SSL refers to a protocol that is used to create an encrypted and authenticated link between a server and a client to enhance the security at this layer.It is a common security technique used by applications.

CVSS Score - 5.9 / Medium

Description

When unpadding an RSA signature, if a client tries to negotiate SSLv2 with a server, that is, set up to support both SSLv2 and more recent SSL and TLS versions, a check is made for a version rollback attack. If they support SSL or TLS versions higher than SSLv2,then they should use special type of padding, that is, it is expected to deny connection requests from a client since this suggests that a version rollback has taken place (i.e. both client and server support greater than SSLv2, and yet this is the version that is being requested).The logic was reversed in the implementation of this padding check, that is, the connection attempt is approved if the padding is there and denied if it is not thus causing security bug.

Exploitation

SSLv2 is supported by OpenSSL 1.0.2. While a client tries to negotiate SSLv2 with a server that is configured to handle both SSLv2 and more modern SSL and TLS versions, a version rollback attack is checked when unpadding an RSA signature. Clients that support SSL or TLS versions higher than SSLv2 are expected to employ a different type of padding. A server that supports more than SSLv2 is expected to reject connection attempts from a client if this special type of padding is present, as this indicates a version rollback.

This padding check's implementation flipped the logic such that the connection attempt is granted if the padding is there and refused if it is not. This implies that if a version rollback attack occurs, such a server will accept a connection. Furthermore, if a normal SSLv2 connection attempt is made, the server will incorrectly reject the connection. This problem affects only OpenSSL 1.0.2 servers from 1.0.2s to 1.0.2x.

To be vulnerable, a 1.0.2 server must:

1. have SSLv2 support enabled at compile time (this is disabled by default),
2. have SSLv2 support enabled at runtime (this is disabled by default), and
3. have SSLv2 ciphersuites enabled. Because OpenSSL 1.1.1 does not support SSLv2, it is not vulnerable to this problem.

The root cause is a bug in the implementation of the RSA_padding_check_SSLv23() function. This also has an impact on the RSA_SSLV23_PADDING padding mode, which is used by a number of other functions.

Although 1.1.1 does not support SSLv2, the RSA_padding_check_SSLv23() method and the RSA_SSLV23_PADDING padding mode are still available. This problem will occur in applications that directly use that method or utilise that padding mode. However, because 1.1.1 lacks compatibility for the SSLv2 protocol, this is considered a bug rather than a security risk.

Impact

1) OpenSSL 1.0.2 servers from version 1.0.2s to 1.0.2x are affected by this issue
2) It will affect the normal connection request as the reverse logic have been implemented

Mitigation

Users should upgrade to 1.1.1j version and premium users should upgrade to 1.0.2y

References

1. https://www.openssl.org/news/vulnerabilities.html
2. https://bugzilla.redhat.com/show_bug.cgi?id=CVE-2021-23839
3. https://www.rapid7.com/db/vulnerabilities/http-openssl-cve-2021-23839/

6.1.2 CVE-2021-20231 - use after free issue in client sending key_share extension may lead to memory corruption

Introduction

This vulnerability was found in GnuTLS.GnuTLS is a secure communications library that implements the SSL, TLS, and DTLS protocols as well as the technologies that surround them. It includes a basic 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.

The project's goal is to create a secure communications back-end that is easy to use and integrates with the rest of the fundamental Linux libraries. A back-end designed to operate and be secure right out of the box, keeping TLS and PKI complexity out of application code.

CVSS Score - 9.8 / Critical

Description

GNUTLS It was found by oss-fuzz that the server sending a "no_renegotiation" alert in an unexpected timing, followed by an invalid second handshake can cause a TLS client to crash via a null-pointer dereference.

Depending on the instantiation and timing of the defect, using previously-freed memory can have a variety of negative repercussions, ranging from the corruption of legitimate data to the execution of arbitrary code. The easiest way for data corruption to occur is for the system to reuse freed memory. There are two typical and occasionally overlapping sources of use-after-free errors:

In this case, the memory in question is legitimately assigned to another pointer after it has been released. The old pointer to the freed memory is reused and refers to a position within the new allocation. As the data is modified, it corrupts the legitimately utilized memory, causing the process to behave erratically. If the newly allocated data contains a class, for example, in C++, numerous function pointers may be spread across the heap data. Execution of arbitrary code is possible if one of these function pointers is replaced with an address to valid shellcode.

Exploitation

It was found by researchers that the client sending a "key_share" or "pre_share_key" extension may result in dereferencing a pointer no longer valid after realloc(). This only happens in TLS 1.3 and only when the client sends a large Client Hello message, e.g., when HRR is sent in a resumed session previously negotiated large FFDHE parameters because the initial allocation of the buffer is large enough without having to call realloc().

Impact

Successful exploitation of these vulnerabilities could lead to the disclosure of sensitive information, addition or modification of data, or Denial of Service (DoS).

Mitigation

It is recommended to upgrade to GnuTLS 3.7.1 or later versions for addressing this issue.

References

1. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-20231
2. https://www.gnutls.org/security-new.html#GNUTLS-SA-2021-03-10
3. https://access.redhat.com/security/cve/cve-2021-20231

6.1.3 CVE-2021-22890: TLS 1.3 session ticket proxy host mixup

Introduction

The vulnerability in curl in establishing TLS 1.3 can cause malicious HTTP Proxy to be used as a Man in the middle attack .

CVSS Score - 2.7 / Low

Description

When confusing the tickets, a HTTPS proxy can trick libcurl to use the wrong session ticket resume for the host and thereby circumvent the server TLS certificate check and make a MITM attack to be possible to perform unnoticed.

Details: When using a HTTPS proxy and TLS 1.3, libcurl can confuse session tickets arriving from the HTTPS proxy but work as if they arrived from the remote server and then wrongly "short-cut" the host handshake. The reason for this confusion is the modified sequence from TLS 1.2 when the session ids would provided only during the TLS handshake, while in TLS 1.3 it happens post hand-shake and the code was not updated to take that changed behavior into account.

When confusing the tickets, a HTTPS proxy can trick libcurl to use the wrong session ticket resume for the host and thereby circumvent the server TLS certificate check and make a MITM attack to be possible to perform unnoticed.

Exploitation

Curl connections maintain two SSL contexts, one for the proxy and one for the destination. However, curl incorrectly stores session tickets issued by an TLS 1.3 HTTPS proxy under the non-proxy context.

The issue is that the logic insideCurl_ssl_addsessionidthat chooses which context to store the tickets under is incorrect under TLS 1.3.

One of the major differences between how TLS session tickets are issued between TLS 1.3 and prior versions of TLS is that TLS 1.3 issues session tickets in a post handshake message. What this means in practice is that TLS 1.3 tickets are delivered in the first call toSSL_read(), rather than being issued as part ofSSL_connect(). Consequently,CONNECT_PROXY_SSL()will see that the proxy has already been connected (since the call toSSL_connect()to the proxy was completed), so the call toCurl_ssl_addsessionidbelieves theisProxyisfalse, and it stores the ticket under the non proxy context.

Impact

This flaw can allow a malicious HTTPS proxy to MITM the traffic. Such a malicious HTTPS proxy needs to provide a certificate that curl will accept for the MITMed server for an attack to work - unless curl has been told to ignore the server certificate check.

Affected versions: curl 7.63.0 to and including 7.75.0

Not affected versions: curl < 7.63.0 and curl >= 7.76.0

Mitigation

Below is the mitigation solution:

Step 1: Upgrade libcurl to version 7.76.0

Step 2: Apply the patch to your local version

Step 3: Use another TLS backend

Step 4: Avoid TLS 1.3 with HTTPS proxies

References

1. https://vulners.com/hackerone/H1:1129529
2. https://curl.se/docs/CVE-2021-22890.html

6.1.4 Researchers report 180K browser extension tampering security headers

Introduction

Browser extensions are client-side add-ons that attempt to improve the functionality of native Web applications. They plan to deliver a rich end-user experience by exploiting feature-rich privileged JavaScript APIs that are otherwise unavailable to native apps. However, multiple large-scale studies have revealed that extensions frequently engage in dangerous activities like ad insertion, collecting privacy-sensitive data, user fingerprinting, spying on user Web activity, and malware dissemination by gaining access to these privileged APIs.

Web applications are frequently the focus of various attacks ranging from Cross-Site Scripting to framing-based attacks and TLS downgrade. Researchers and practitioners have developed many mitigations for these attacks, which are generally provided by the server through HTTP headers and then enforced by the client. For example, to regulate script inclusion and framing from third-party pages, the server may establish a Content-Security-Policy header. However, given the aforementioned capabilities, extensions may edit or remove such headers, thereby disabling well-configured security safeguards and compromising the application's security, which is meant to be enforced by the client.

The Web server provides HTTP security headers and transmits them with the response so that the proper security protocol is enforced at the client by the browser. Tampering with these headers may jeopardize the apps' client-side security and expose them to vulnerabilities that the headers aim to mitigate. Over 186K publicly available Chrome browser extensions were investigated for studying this behavior. Numerous research studies have consistently revealed the usage of these extensions as vectors to carry out harmful actions, potentially compromising privacy and security. Previous large-scale investigations by numerous academics show that they frequently spy on user browsing history, steal sensitive user information, or illegally modify the content of Web sites.

CVSS Score - 7 / High

Description

186K Chrome extensions, publicly available on the Chrome Web Store, to detect extensions that actively intercept requests and responses and tamper with their security headers by either injecting, dropping, or modifying them, thereby undermining the security guarantees that these headers typically provide. The impact of this was exposing consumers to a wide range of web-based attacks. While security headers are a little-known technical element, they are a crucial aspect of today's internet ecosystem.

A security header is a technical term for an HTTP response provided by the server to a client app, such as a browser. When a user visits a website, the browser sends a request to the server, which then returns the webpage. While websites are presented using HTML, JavaScript, and CSS code, website managers can include extra parameters in the HTTP connection header to direct the user's browser on how to treat the given material. Security headers are a sort of HTTP response that internet standards bodies have developed over time to allow website owners to activate and adjust security measures within the user's browser or other client apps. Some of the most popular security headers used today are often used by website administrators to ensure that their site operates over an encrypted HTTPS connection, that users are protected from cross-site scripting attacks, and that code running inside iframes cannot steal their browser data.

Using a custom framework they built specifically for a study, the research team said they analyzed 186,434 Chrome extensions that were available on the official Chrome Web Store last year.

Exploitation

While website administrators specify their security headers, this does not imply they are still present on the client-side, where they can be intercepted and deactivated by attackers using a man-in-the-middle attack, malware running on an operating system, or browser extensions.

However, while the first two categories are obvious attack vectors and malicious in nature, the third is more difficult to describe because, while some extensions may be malicious, other extensions may disable security headers for more benign reasons, such as enabling a certain functionality blocked by the header or by mistake.

The work found that 2,485 extensions were intercepting and modifying at least one security header used by today’s Top 100 most popular websites (Source - Tranco list).

Category	# Extensions
Extensions analysed in this phase	186,434
Extensions with target API Privileges	17,434
Extensions with relevant function signatures	3,286

Extensions targeting <all_urls> & wildcards	2,694
Extensions targeting specific hosts	592
No. of distinct domains extracted	4,550

I. Summary of the findings from static analysis

Category	# Extensions
Extensions analyzed in this phase	3,286
No. of distinct domains crawled on	4,637
Extensions that modify at least one security header	2,685
Extensions that modifies all four security headers	553
Extensions that target <all_urls> & wildcards	2,245
Extensions that target specific hosts	240

II. Summary of the findings from runtime analysis

The study didn't focus on all security headers, but only on the four most common ones, such as Content-Security-Policy (CSP), HTTP Strict-Transport-Security (HSTS), X-Frame-Options, and X-Content-Type-Options.

While 2,485 extensions deactivated at least one, researchers discovered 553 that disabled all four security headers examined in the research.

CSP, a security header intended to let site owners regulate what online resources a page is permitted to load inside a browser and typical protection that helps defend websites and browsers from XSS and data injection threats, was the most widely disabled security header.

Target Security Headers	Extensions Involved in		
	Alteration	Injection	Dropping
Content-Security-Policy	1,412	-	509
Content-Security-Policy-Report-Only	1,436	-	1,436
Strict-Transport-Security	1,338	844	679
X-Frame-Options	1,345	811	676
X-Content-Type-Options	1,115	707	553

III. Different security headers & their instances of manipulation by extensions

Impact

According to the study team, the Chrome extensions disabled CSP and other security headers "to introduce extra seemingly innocuous functions on the visiting webpage" and did not appear to be malicious in nature in the majority of the situations they examined.

Even though the extensions intended to improve a user's online experience, German scholars said that by interfering with security headers, the extensions only exposed users to assaults from other scripts and sites operating inside the browser and on the web.

It is imperative to quantify the risks of other pertinent headers as well, such as Referrer-Policy which permits Web servers to control referrer information sent along with the request, or CORS headers which regulate resource-sharing across different origins. If injected by an extension, this could potentially allow for access to cross-domain resources, which would usually be protected through the Same-Origin Policy.

Mitigation

Not all extensions are what they claim to be. Chrome Web Store doesn’t entirely make it impossible to install crooked programs. However, it remains the most secure platform to install extensions. Getting your add-ons from the official web store doesn’t in any way invalidate other best practices in this article. Therefore, give them equal consideration.

- Choose Reliable Developers
- Consider Product Rating
- Be Conscious of Privileges You Grant Extensions
- User Training

References

1. https://swag.cispa.saarland/papers/agarwal2021extensions.pdf

6.1.5 ALPACA (Application Layer Protocol Content Confusion Attack) Attack

Introduction

TLS is the data protection standard that places a padlock in your browser's address bar, encrypts your email while it's being transferred (hopefully), and prevents cybercriminals from inadvertently replacing software you download with malware and other nasties.

The TLS protocol operates as follows:

- To safeguard your data from prying and monitoring, agree on a one-time encryption key with the opposite end of the connection.
- Making it more difficult for fraudsters to put up bogus sites to fool you by verifying the person or company controlling the server on the other end.
- Checking the integrity of data as it comes to prevent other network users from interfering with the information.

TLS secures network communications by adding an additional layer of authentication to numerous application layer protocols such as HTTPS, SMTP, IMAP, POP3, and FTP. The TLS protocol's ultimate purpose is not merely encryption; it is intended to provide the communication between the source and the destination with integrity, confidentiality, availability, and nonrepudiation.

Given how much we rely on TLS, anytime a vulnerability is revealed in it, the disclosure usually makes significant headlines. Perhaps amusingly, this has had a kind of virtuous circle effect, with researchers going out of their way to come up with names and logos for TLS vulnerabilities that promote large headlines in the first place.

We jokingly refer to them as BWAINs - an impressive name that's short for bug with an amazing name – and examples include BEAST, Heartbleed, Logjam, Lucky Thirteen, and now...the amusingly titled ALPACA.

CVSS Score - 8.3 / High

Description

The application layer protocol content confusion attack (ALPACA) was initially revealed in June and demonstrated at Black Hat USA 2021. Understanding ALPACA requires an understanding of how TLS works: The protocol is meant to safeguard data in transit during a transaction, but it does not bind TCP connections to the intended application layer protocol, which may be HTTP, SMTP, or any of the many other protocols frequently protected with TLS. In reality, this implies that while TLS protects data as it travels and checks the server name to which it connects, it does not examine the application to whom the data is being delivered or even the veracity of that data.

Threat actors can exploit the aforementioned in an ALPACA attack by utilizing man-in-the-middle tactics to redirect a victim's browser to a different FTP, POP3, or another service that has the same certificate. The attacker dupes the browser into transferring sensitive data to the other server. The attacker can then extract session cookies and other sensitive data, as well as run arbitrary JavaScript on the web browser.

ALPACA attacks demonstrate how traffic redirection may be used to circumvent security both within and outside your network, rather than merely creating interruption or denial of service, as you might expect at first.

TLS protects the raw data sent over the connection it's protecting and checks the name of the server it's been asked to connect to, but it doesn't formally validate the application running at each end of the link or evaluate the legitimacy of the data being transferred.

In other words, in an ALPACA attack, the padlock would appear in your browser, you'd be uninformed that you weren't connecting to the server you anticipated, and your browser would innocently and trustingly begin communicating with another server on the network instead.

According to the researchers, 1.4 million web servers were vulnerable to these cross-protocol assaults. 119,000 of these web servers might be compromised using vulnerable application servers.

Exploitation

The image depicts three different methods for an attacker to conduct cross-protocol assaults on webservers by abusing insecure FTP and Email servers: The attacker uses the Upload Attack to steal authentication cookies or other secret information. The attacker uses the Download Attack to launch a stored XSS attack. The attacker uses the Reflection Attack to execute a reflected XSS in the context of the victim's website.

- Upload Attack – attacker exfiltrates authentication cookies or other private data

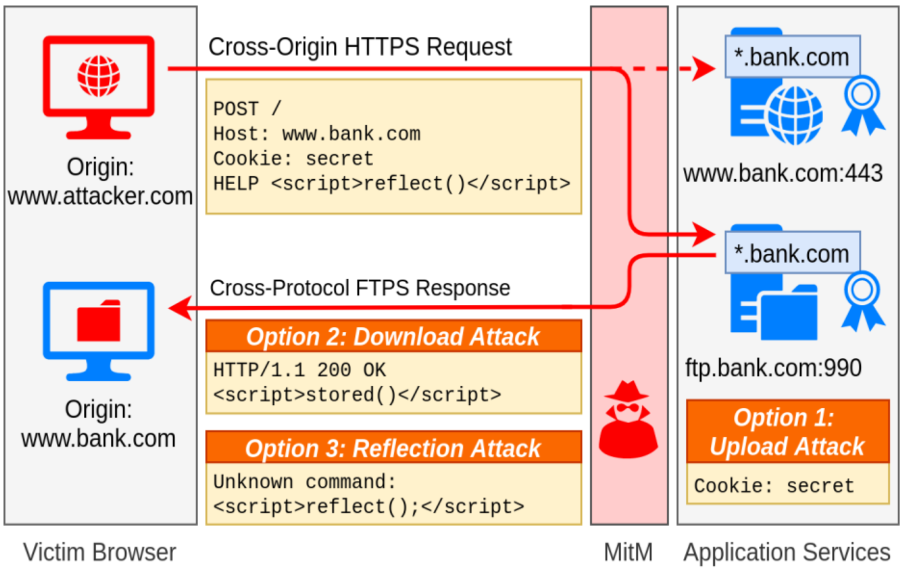
- Download Attack – attacker executes a stored cross-site scripting (XSS) attack
- Reflection Attack – attacker executes a reflected XSS in the context of the victim's website

XSS is a huge web security hole because the reflected script can access data such as login cookies specific to the site you're currently visiting, and thereby steal your login, raid your shopping cart, or otherwise poke its nose into your online business. Email servers, on the other hand, don't generally deal with JavaScript, and their responses are supposed to make sense to email-sending applications, so pointing a browser at a mail server and sending a carefully crafted but bogus web request may cause the email server to produce, among other things, an error message that hasn't been subjected to the same rigorous anti-XSS checking as a web server.

"So what?" you're probably thinking again. What harm would it cause if the email server returned some rogue, mirrored JavaScript? An attacker would go nowhere since there are no session cookies, shopping carts, or other private online data linked with the email server."

Except for one thing: the browser believes it is connecting to the genuine web server, and it believes this because it was supplied with a TLS certificate that would have been valid for the webserver if it had ended up there.

As a result, even if the browser did not connect to the web server, the rogue script mirrored by the well-meaning email server would be able to read out the browser cookies and web data connected with the web server.



```
POST /
Host: www.bank.com
Cookie: secret
HELP <script>reflect()</script>
```

```
HTTP/1.1 200 OK
<script>stored()</script>
```

```
Unknown command:
<script>reflect();</script>
```

```
Cookie: secret
```

The researchers reevaluated Jann Horn's two assaults in their ALPACA attack and included a third scenario in which the attacker sends the victim's secret cookie to the FTP server. In addition to FTP, they investigated whether these attacks could be carried out using the email protocols SMTP, IMAP, and POP3. They examined 24 FTP and email servers and discovered that 12 of them enabled at least one browser attack.

Impact

All of this begs the question, "How could a browser in the first place mix up a web server's TLS certificate with an email server's certificate?"

Until certificate-issuing organizations like Let's Encrypt came along and simplified the process of getting TLS certificates both cheap and simple, buying and upgrading certificates for all the servers on your network was normally a bit of a pain (and cost).

As a result, businesses rightfully rely on certificates that are valid for many, many, or even all of the servers in their network domain.

Instead of getting a separate certificate for, say, www.example.com and mail.example.com, you could use a wildcard certificate that's valid for *.example.com, where the asterisk (star) character denotes "any name at all," just as most file-finding programs interpret *.DOCX as "all files that end with a DOCX extension."

And there, oversimplified is the crux of the ALPACA issue.

TLS certificates valid for more than one type of server on your network might be used to carry out the CA element of ALPACA, specifically the Cross-protocol Attacks.

Your browser ends up trusting the incorrect server and communicating with it in the wrong language, yet it is still capable of doing some type of destructive security bypass without physically hacking any of the servers.

The following is a list of analyzed implementations in terms of vulnerability:

- When used via STARTTLS, Sendmail SMTP permitted reflection attacks that work in Internet Explorer.
- IMAP servers from Cyrus, Kerio Connect, and Zimbra enabled to download and reflection attacks that work on Internet Explorer.
- Courier, Cyrus, Kerio Connect, and Zimbra all supported download assaults in Internet Explorer.

- Reflection attacks that work in Internet Explorer are supported by Microsoft IIS, vsftpd, FileZilla Server, and Serv-U FTP servers. Furthermore, the same FTP hosts were permitted to upload and download assaults that worked in all browsers.

Even still, there are interactions with both studied and unanalyzed procedures, making risk estimate problematic because it is believed that there are a vast number of yet unidentified vulnerabilities in this domain.

Browsers are often impacted by the vulnerability, however, they are not to blame. It was also discovered that some browsers are more susceptible than others due to how they handle non-HTTP answers.

Mitigation

If you're concerned about visitors to your network being misled by an admittedly implausible ALPACA attack, the researchers have discovered numerous strategies to limit the likelihood of this type of TLS misuse.

1. Use Application-Level Hardening
2. Avoid TLS Certificate Overlap - Wildcard certificates are widely used and useful for network administrators that manage dozens or hundreds of distinct subdomains on a company network. Nonetheless, if possible, avoid wildcards and confine each certificate such that it only vouches for a list of server names related to a certain service or collection of services. Instead of receiving a certificate for *.example.com that all of your web servers and SMTP servers may use, try getting one certificate for each type of server and designating the appropriate servers specifically in each one.
3. Use Application Layer Protocol Negotiation (ALPN) if you can - The ALPN extension enables the client to transmit a list of intended protocols, and the server to choose one of these protocols or terminate the connection. In this manner, any attempt to divert traffic from a server using a different protocol would be foiled.
4. Use Server Name Indication (SNI) if you can - In addition to ALPN, thorough validation of the Server Name Indication (SNI) can provide further defense-in-depth against any sort of cross-host attack.

References

- https://alpaca-attack.com/
- https://alpaca-attack.com/ALPACA.pdf
- https://github.com/RUB-NDS/alpaca-code/
- https://blog.apnic.net/2021/11/10/alpaca-attack-analysing-and-mitigating-cracks-in-tls-authentication/
- https://nakedsecurity.sophos.com/2021/06/11/alpaca-the-wacky-tls-security-vulnerability-with-a-funky-name/

6.1.6 CVE-2021-38165 Lynx exposes authentication credentials over TLS

Introduction

Lynx is a text-only web browser. Lynx is a terminal-based web browser that is available for all Linux variants. On the terminal, the result is shown as plain text. It's a traditional non-graphical text-mode web browser that shows websites on the terminal. Lynx does not load graphics or multimedia material, hence it is faster than other browsers. Those that do not want a lot of photos and multimedia on their web pages prefer to use Lynx. It is simple to install and operate, and it does not require any additional interface to execute or display the contents; instead, it does it via the terminal.

Javascript and Flash content is not supported on Lynx, but it supports the tracking cookies. It provides users with the ability to maintain greater privacy by disabling the use of cookies if they wish to do so. It was originally designed for use on Unix-like systems, it is also available for Windows and MacOS.

CVSS Score - 5.3 / Medium

Description

In Lynx version, before 2.8.9, the userinfo part of a URL was mishandled, because of which the credentials may appear in SNI data. During the TLS connection handshake these credentials were included in the Server Name Indication (SNI) TLS extension data unencrypted and were sent. If the attacker eavesdrops on the network connection between lynx browser and server, he/she will be able to get user authentication credentials.

Exploitation

If the attacker and user are on the same network, the attacker can intercept or eavesdrop into the requests sent by the user and get over the credentials as they are transmitted in the plain text format without any encryption. The credentials will be visible to us if we use any of the sniffer tool available in the industry, for eg. WireShark.

Due to TLS Server Name Indication (SNI) the hostname as parsed by Lynx (i.e with "user:pass@" included) is sent in _clear_ text over the wire even _before_ an user can even said "n" for "no, don't continue to talk with this server" in Lynx's prompt.

It was possible to capture the password given on the commandline in traffic of an TLS handshake using tcpdump and analysing it with Wireshark:

```
From Wiresharks TLS dissector:


Server Name Indication extension
    Server Name list length: 28
    Server Name Type: host_name (0)
    Server Name length: 25
    Server Name: user:pass@www.example.org
                          ^^^^^^^^^^


From Wiresharks "Follow TCP stream":


..........a
....jV.. ......../.......D.&....R.+.,.....      .
.../.0..............z.{./.5.A...
.....|.}.3.9.E............2.8.D.......p...........$."...user:pass@www.example.org......#...
...
................
.............................
```

Impact

1) Leakage of User Authentication Credentials.
2) Lynx version 2.8.9 have been been affected from this vulnerability
3) The data availability may get affected if the attacker changed the credentials which is a serious concern

Mitigation

1) Users should use Multi Factor Authentication.

2) Users should upgrade to version 2.8.9-4 of Lynx

References

1. https://access.redhat.com/security/cve/cve-2021-38165
2. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-38165
3. https://www.debian.org/security/2021/dsa-4953
4. https://security-tracker.debian.org/tracker/source-package/lynx

6.1.7 CVE-2021-22946 A bug in CURL leads to Information disclosure

Introduction

Data transfer to and from a server necessitates the use of tools that support the relevant network protocols. Curl and wget are two of the most used Linux utilities for this purpose.

Curl (short for "Client URL") is a command-line program for transferring data over several network protocols. It talks with a web or application server by giving a URL and the data to be transmitted or received.

libcurl, a portable client-side URL transfer library, powers curl. It may be used straight from the command line or as part of a script. Curl's most prevalent applications are:

- Downloading files from the internet
- Endpoint testing
- Debugging
- Error logging

TLS (Transport Layer Security) is the authentication and encryption successor to SSL. TLS is a cryptographic protocol that is used to improve the security of network communication. TLS is mostly used to secure communication between web and mobile apps and the webserver.

Curl has support for Secure Transport connections (its more secure version is called TLS). When you make a Curl request for an HTTPS URL, Curl validates the SSL certificate for the destination URL against the local CA certificate store and alerts you if it is invalid, self-signed, or has expired. This is ideal for production websites but inconvenient for development.

Information disclosure, also known as information leakage, is when a website unintentionally reveals sensitive information to its users. Depending on the context, websites may leak all kinds of information to a potential attacker, including:

- Data about other users, such as usernames or financial information
- Sensitive commercial or business data
- Technical details about the website and its infrastructure

CVSS Score - 7.5 / High

Description

A needed TLS bypassing problem in cURL ***libcurl*** allowed a remote attacker to get sensitive information. An attacker might exploit this vulnerability by sniffing the network to gather sensitive data in plain text across the network and use this information to perform further attacks against the vulnerable machine.

This vulnerability exists in the curl versions >= 7.20.0 and <= 7.78.0.

The vulnerability arises as a result of an issue linked to inappropriate command-line enforcement of the *--ssl-reqd* option or *CURLOPT USE SSL* setting set to *CURLUSESSL CONTROL* or *CURLUSESSL ALL* with ***libcurl***. A remote attacker with control over the IMAP, POP3, or FTP server can deliver a specially designed but perfectly genuine answer to the ***libcurl*** client, forcing curl to silently continue its activities without TLS, despite the instructions and expectations, potentially exposing sensitive data in clear text across the network.

In simple words, in IMAP and POP3, *--ssl-reqd* is silently ignored if the capability command failed. In FTP, a non-standard 230 response in the greeter message forces curl to continue unencrypted, even if TLS has been required.

Exploitation

Steps to reproduce: Using a parameterizable test server to fail capability command for IMAP (CAPABILITY reply: A001 BAD Not implemented) and POP3 (CAPA reply: -ERR Not implemented) and to send response code 230 in FTP server greeting message.

1. curl --ssl-reqd imap://server/...
2. curl --ssl-reqd pop3://server/...
3. curl --ssl-reqd --ftp-ssl-control ftp://server/…

These 3 commands were successful, but network sniffing showed that TLS was never negotiated.

Impact

A MitM (Man-in-the-Middle) can discreetly refuse necessary TLS negotiation, allowing it to sniff and/or change unencrypted data sent.

Mitigation

A fix for this vulnerability can be found in this GitHub commit.

The following recommendations are suggested by the Curl Project:

- Upgrade curl to version 7.79.0
- Apply the patch to your local version
- Do not use IMAP, POP3, or FTP

References

1. https://curl.se/docs/CVE-2021-22946.html
2. https://exchange.xforce.ibmcloud.com/vulnerabilities/209452
3. https://hackerone.com/reports/1334111

6.1.8 CVE-2021-22947 A bug in CURL affects data integrity

Introduction

Data transfer to and from a server necessitates the use of tools that support the relevant network protocols. Curl and wget are two of the most used Linux utilities for this purpose.

Curl (short for "Client URL") is a command-line program for transferring data over several network protocols. It talks with a web or application server by giving a URL and the data to be transmitted or received.

libcurl, a portable client-side URL transfer library, powers curl. It may be used straight from the command line or as part of a script. Curl's most prevalent applications are:

- Downloading files from the internet
- Endpoint testing
- Debugging
- Error logging

TLS (Transport Layer Security) is the authentication and encryption successor to SSL. TLS is a cryptographic protocol that is used to improve the security of network communication. TLS is mostly used to secure communication between web and mobile apps and the webserver.

Curl has support for Secure Transport connections (its more secure version is called TLS). When you make a Curl request for an HTTPS URL, Curl validates the SSL certificate for the destination URL against the local CA certificate store and alerts you if it is invalid, self-signed, or has expired. This is ideal for production websites but inconvenient for development.

Data integrity is a concept and procedure that assures an organization's data is accurate, full, consistent, and legitimate. Organizations who follow the method not only assure the integrity of the data but also that they have accurate and correct data in their database.

As data quantities continue to grow dramatically, the relevance of data integrity grows. To forecast customer behavior, monitor market activity, and reduce possible data security threats, major enterprises are growing more reliant on data integration and the capacity to properly understand information. This is critical for data mining since it allows data scientists to work with the correct information.

CVSS Score - 5.9 / Medium

Description

When connecting to an IMAP, POP3, SMTP, or FTP server to exchange data securely using STARTTLS to upgrade the connection to TLS level, cURL ***libcurl*** is vulnerable to a man-in-the-middle attack due to weakness. An attacker might take advantage of this flaw to execute a man-in-the-middle attack and get access to the communication channel between endpoints in order to collect sensitive information or further damage the system.

Curl >= 7.20.0 and <= 7.78.0 may connect to an IMAP or POP3 server to get data using STARTTLS to upgrade to TLS security, and the server can react and send back several responses at once, which curl caches. Curl would then upgrade to TLS without flushing the in-queue of cached replies, instead continuing to use and trust the responses received **before** the TLS handshake as though they were authenticated. Using this issue, a Man-In-The-Middle attacker may inject phony answers, then pass through TLS traffic from the real server and mislead curl into sending data back to the user, believing the attacker's injected data is from the TLS-protected server.

Exploitation

You can use the following test case within the curl test system. It is based on IMAP FETCH with explicit TLS. Upon test failure, the downloaded file contains "You've been hacked!" rather than the requested mail.

Test case:

```
<testcase>

<info>

<keywords>

IMAP

STARTTLS

</keywords>

</info>


#

# Server-side

<reply>
```

```
<servercmd>

CAPA STARTTLS

REPLY STARTTLS A002 BAD currently unavailable\r\nA003 OK Authenticated\r\nA004 OK
Selected\r\n* 1 FETCH (BODY[TEXT] {21}\r\nYou've been hacked!\r\n)\r\nA005 OK
Completed

</servercmd>

<data nocheck="yes">

From: me@somewhere^M

To: fake@nowhere^M

^M

body^M

^M

--^M

  yours sincerely^M

</data>

</reply>


#

# Client-side

<client>

<features>

SSL

</features>

<server>

imap

</server>

 <name>

IMAP STARTTLS pipelined server response

 </name>
 <command>
'imap://%HOSTIP:%IMAPPORT/%TESTNUMBER/;MAILINDEX=1' -u user:secret --ssl
</command>
</client>


#
# Verify data after the test has been "shot"
<verify>
# 8 is CURLE_WEIRD_SERVER_REPLY
<errorcode>
8
</errorcode>
<protocol>
A001 CAPABILITY
A002 STARTTLS
</protocol>
</verify>
</testcase>
```

Impact

Using this issue, a Man-In-The-Middle attacker may inject phoney answers, then pass-through TLS traffic from the real server and mislead curl into sending data back to the user, believing the attacker's injected data is from the TLS-protected server.

An attacker can insert fraudulent response data over POP3 and IMAP.

Mitigation

A fix for this vulnerability can be found in this GitHub commit.

The following recommendations are suggested by the Curl Project:

- Upgrade curl to version 7.79.0 or the latest version.
- Apply the patch to your local version
- Use IMAPS://, POP3S://, SMTPS:// or FTPS:// with implicit TLS instead of using STARTTLS

References

- https://curl.se/docs/CVE-2021-22947.html
- https://hackerone.com/reports/1334763
- https://www.ibm.com/support/pages/security-bulletin-security-vulnerabilities-affect-ibm-cloud-private-curl-cve-2021-22947

6.1.9 CVE-2021-4044 Invalid handling of X509_verify_cert() internal errors in libssl

Introduction

The LibSSL project is a part of the OpenSSL protocol which helps in the implementation of SSL and TLS for secure communication over the internet.

X509 build chain() constructs a certificate chain beginning with target and ending with the optional list of intermediate CA certificate certs. If the store is NULL, the chain is built as far down as feasible while disregarding failures. Otherwise, the chain must reach a trust anchor in the store. It employs an X509 STORE CTX structure that is linked to the library context libctx and the property query string propq, both of which might be NULL. If there are several possibilities for the chain, only one is chosen.

On success, it delivers a reference to a new stack of certificates, beginning with target and progressing through all available intermediate certificates. Only if the target is the trust anchor of with self signed is a self-signed trust anchor included. If a non-NULL stack is returned, it is the caller's responsibility to release it.

Based on the arguments in ctx, the X509 verify cert() function attempts to identify and validate a certificate chain.

CVSS Score - 7.5 / High

Description

This vulnerability was present in OpenSSL 3.0.0 version.It occurs because of the X509_verify_cert() call on the client side for the verification of certificate supplied by the server. Calling this function may return a negative value indicating an internal error. And the problem starts here. This negative value is mishandled by the OpenSSL causing an I/O function indicating the failure and subsequently a call will be made to SSL_get_error() to return SSL_ERROR_WANT_RETRY_VERIFY as a value.But this value should only be returned in the case where application has called previously the SSL_CTX_set_cert_verify_callback().

Exploitation

To validate a certificate issued by a server, libssl in OpenSSL uses X509 verify cert() on the client side. A negative value may be returned by the function to indicate an internal error (for example out of memory). OpenSSL mishandles such a negative return result, causing an IO function (such as SSL connect() or SSL do handshake()) to fail and a subsequent call to SSL get error() to return the value SSL ERROR WANT RETRY VERIFY. OpenSSL is only meant to return this value if the application has already invoked SSL CTX set cert verify callback (). Because most applications do not do this, the SSL ERROR WANT RETRY VERIFY return value from SSL get error() will be completely unexpected, causing applications to behave incorrectly.

The precise behaviour will vary depending on the programme, but it may result in crashes, infinite loops, or other inappropriate answers. This problem is exacerbated by a second fault in OpenSSL 3.0 that causes X509 verify cert() to report an internal error while processing a certificate chain. This happens when a certificate lacks the Subject Alternative Name extension yet a Certificate Authority has imposed name limits. Even with proper chains, this problem can arise. An attacker could cause incorrect, application-dependent behaviour by combining the two issues.

Impact

Succesfull exploitation of this vulnerability may lead to crashing of application i.e. leading to Denial of Service (DoS).

Users of the following products have been affected:-

1) NetApp HCI Baseboard Management Controller (BMC) - H300S/H500S/H700S/H410S
2) FAS/AFF Baseboard Management Controller (BMC) - A250/500f
3) NetApp HCI Baseboard Management Controller (BMC) - H410C
4) ONTAP Select Deploy administration utility

Mitigation

Users should upgrade to 3.0.1 version of OpenSSL

References

1. https://www.openssl.org/news/secadv/20211214.txt
2. https://bugzilla.redhat.com/show_bug.cgi?id=CVE-2021-4044
3. https://www.cve.org/CVERecord?id=CVE-2021-4044
4. https://www.suse.com/security/cve/CVE-2021-4044.html

6.2 Certificates related flaws

6.2.1 Microsoft Exchange Admin Portal Goes Down Due to an Expired SSL Certificate

Introduction

The Microsoft Exchange Admin Center (EAC) is a contemporary, web-based administration dashboard for administering Exchange that is intended to give a more consistent admin experience with the rest of Microsoft 365. It takes the role of the Exchange Control Panel (ECP) in managing an organization's email settings.

The EAC brings some confounding features like a Personalized dashboard, Azure Cloud Shell Support, Migration capabilities, and more.

Have you ever noticed how some URLs begin with http:// and others begin with https://? Perhaps you noticed the extra "s" while you were visiting websites that required you to enter sensitive information, such as when you paid bills online. But where did that additional "s" come from, and what does it mean?

Simply said, the additional "s" indicates that your connection to that website is secure and encrypted and that any data you provide is safely exchanged with that website. SSL, which stands for "Secure Sockets Layer," is the technology that enables the little "s."

To create this secure connection, an SSL certificate (also referred to as a "digital certificate") is installed on a web server and serves two functions:

- It authenticates the identity of the website (this guarantees visitors that they're not on a bogus site).
- It encrypts the data that's being transmitted.

SSL certificates have a validity term, similar to how a driver's license must be updated in order to retain correct and up-to-date information about your identification. When this time period expires, browsers show a notice on the webpage, indicating that the SSL certificate has expired. You will no longer be able to interact over a secure, encrypted HTTPS connection when an SSL certificate expires. All information will be transferred unencrypted, leaving the data vulnerable to any network attacker. This is why it is critical to understand the validity term of an SSL certificate used by any firm.

CVSS Score - 6 / Medium

Description

Microsoft Exchange administrators who intended to access the admin.exchange.microsoft.com awoke to warnings on their browsers when they attempted to enter the Microsoft Exchange Admin Portal on May 23, 2021, a Sunday,   The portal became unreachable with the warning "Your connection is not private," as a result of an outage caused by an expired SSL certificate. According to Qualys SSL Labs, the certificate linked with the Exchange Admin Portal expired at 8 a.m. ET on Sunday, which eluded Microsoft's notice.

Encrypted communications increase complexity and human mistake, such as forgetting to renew an SSL certificate. Certificate expiration causes disruptions, which are becoming all too regular now that the great majority of internet services have converted to secure connections. According to a survey done in the United Kingdom, human error is the principal source of the majority of cyberattacks, accounting for 60% of them. The employees are the first line of defense against hackers, and implementing procedures is simply the first step in ensuring they respond appropriately to occurrences.

The expiration of a single certificate is sometimes enough to stop an application or even an entire infrastructure from working. This episode, however, should not come as a surprise. It is not the first time that Microsoft has failed to renew an expired certificate; in February 2020, Teams collapsed for 3 hours due to a failure to renew the associated certificate.

Many firms find it difficult to manage SSL certificates. Despite the importance of digital certificates for brand reputation, service dependability, and availability, certificates do expire due to poor management practices. This was the situation with both Spotify and the State of California, where an expired certificate resulted in a backlog of roughly 300,000 lab reports in the state's coronavirus reporting system.

Exploitation

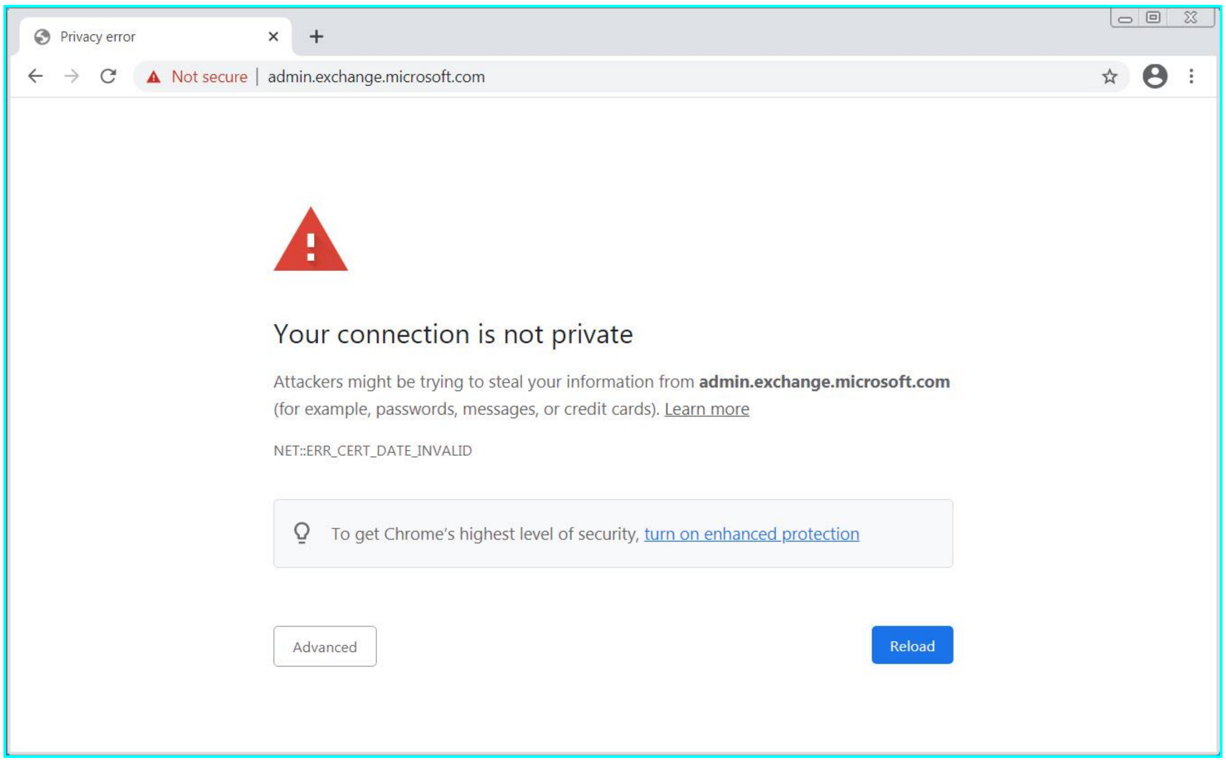
The Microsoft Exchange admin interface was unavailable from some browsers due to Microsoft's failure to renew the website's SSL certificate.

Beginning around 8 a.m. EST on a Sunday, Microsoft Exchange administrators who attempted to enter the admin portal at admin.exchange.microsoft.com discovered that their browsers were notifying them that the connection was not private owing to an expired SSL certificate.

Users were either prohibited from visiting the site as a security precaution or provided a message that the data may not be safe, depending on the browser. Google Chrome, for example, prevented the users from viewing the site entirely, whilst Firefox was notifying users about the unsafe connection.

While Microsoft hustled to fix the issue, Tzatl was quick to catch up with the outage. One of the tweets read, “What’s going in with http://admin.exchange.microsoft.com? Did you guys really forget to renew a certificate?”. Microsoft reacted to this tweet by indicating that they had identified the issue and were working to resolve it. In the meanwhile, the business encouraged consumers to the service health dashboard for further information. Later, Microsoft reissued the failing certificate and quickly restored access to the portal.

Expired SSL Certificate for admin.exchange.microsoft.com

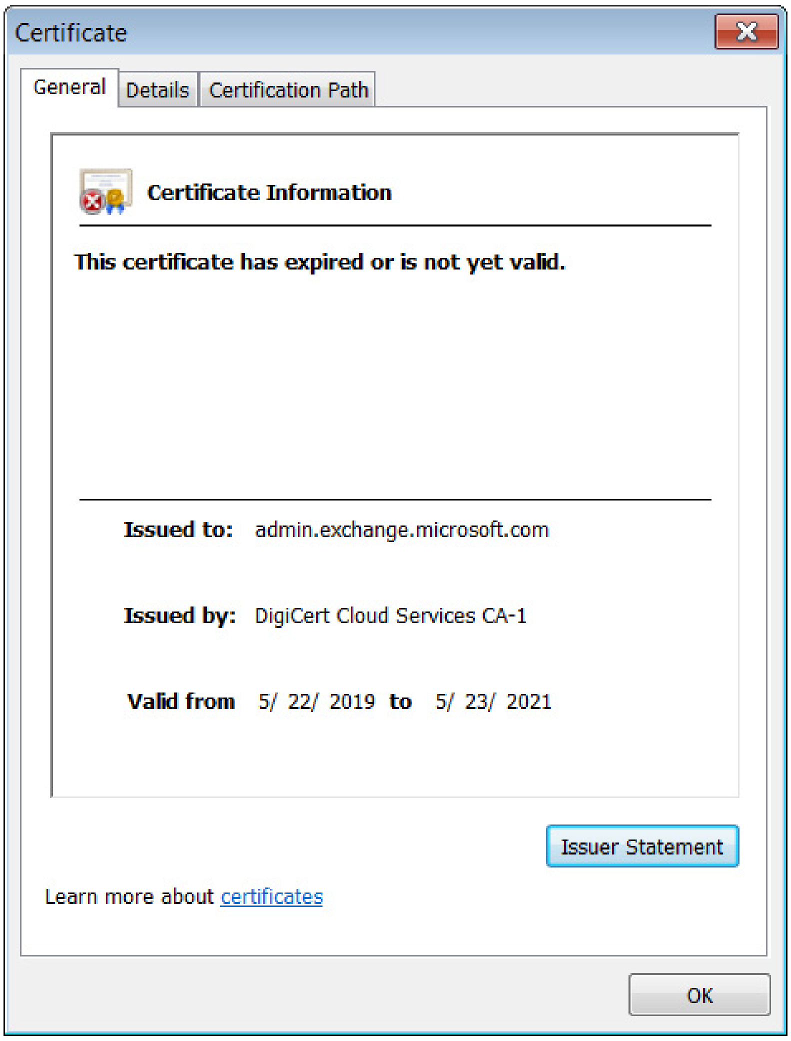


Your connection is not private

Attackers might be trying to steal your information from **admin.exchange.microsoft.com** (for example, passwords, messages, or credit cards). Learn more

NET::ERR_CERT_DATE_INVALID

To get Chrome’s highest level of security, turn on enhanced protection


Certificate

General | Details | Certification Path

Certificate Information

This certificate has expired or is not yet valid.

Issued to: admin.exchange.microsoft.com

Issued by: DigiCert Cloud Services CA-1

Valid from 5/ 22/ 2019 to 5/ 23/ 2021

Impact

SSL certificates are much more than a nice-to-have feature for your website. In fact, they have become a need if you want to run a profitable website that is secure for your users. While using an SSL certificate is not required, the rate at which unencrypted online traffic is intercepted and users' PCs and web servers are hacked is disturbing, to say the least.

At first glance, the event may appear inconsequential, given that the repercussions of the outage were minor, resulting in just a few hours of downtime (however Gartner indicates that the average cost of IT downtime is close to $5,600 per minute!). It could even be easier to dismiss it as a small operational glitch. However, the large-scale security breaches at Equifax and SolarWinds, the ramifications of which are still being felt, serve as sharp reminders of what certificate expirations may do. As a result, it is only reasonable to not disregard the significant security risk that Microsoft made by failing to renew the relevant certificate on time.

Mitigation

The first step is to create and regularly update a complete certificate inventory. Following that, expiry notifications should be put up to guarantee that the proper owners are notified ahead of time. This comprises a fixed time beginning at least one month before the expiry date for non-essential systems and beginning at least two months before the expiry date for critical systems.

Automation of digital certificate procedures such as discovery, enrollment, renewals, provisioning, and revocation aids in the elimination of human mistakes and neglect, which are major causes of certificate vulnerabilities. When a certificate's expiration date approaches, a certificate lifecycle management solution with native automation automatically obtains a certificate from the CA by producing a new CSR or reusing an existing CSR and installing it on the endpoint.

References

1. https://www.appviewx.com/blogs/microsoft-exchange-admin-portal-goes-down-due-to-an-expired-ssl-certificate/
2. https://hitechglitz.com/microsoft-exchange-management-portal-blocked-by-an-expired-ssl-certificate/

6.2.2 Certificate Validation error in BIOS of DELL products

Introduction

Eclypsium researchers have identified multiple vulnerabilities affecting the BIOSConnect feature within Dell Client BIOS. This chain of vulnerabilities has a cumulative CVSS score of 8.3 (High) because it allows a privileged network adversary to impersonate Dell.com and gain arbitrary code execution at the BIOS/UEFI level of the affected device. Such an attack would enable adversaries to control the device’s boot process and subvert the operating system and higher-layer security controls. The issue affects 129 Dell models of consumer and business laptops, desktops, and tablets, including devices protected by Secure Boot and Dell Secured-core PCs.

The Eclypsium team has coordinated with Dell PSIRT throughout the disclosure process. Dell has issued a Dell Security Advisory and released BIOS/UEFI updates for affected systems and updates to affected executables from Dell.com.

These vulnerabilities enable an attacker to remotely execute code in the pre-boot environment. Such code may alter the initial state of an operating system, violating common assumptions on the hardware/firmware layers and breaking OS-level security controls. As attackers increasingly shift their focus to vendor supply chains and system firmware, it is more important than ever that organisations have independent visibility and control over the integrity of their devices..

CVSS Score - 8.3 / High

Description

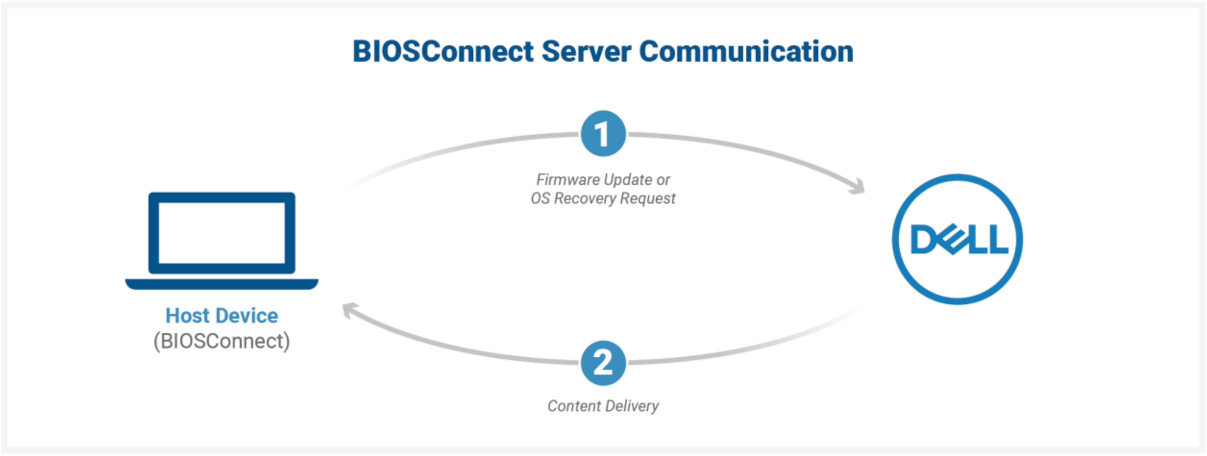
Dell SupportAssist is an overarching support solution that comes preinstalled on most Windows-based Dell machines. SupportAssist covers a range of support functions such as monitoring for hardware and software problems and assisting with troubleshooting and recovery when issues are found.

BIOSConnect is a feature of SupportAssist that allows users to perform a remote OS recovery or update the firmware on the device. In either case (firmware update or OS recovery), BIOSConnect enables the system’s BIOS to reach out to Dell backend services over the Internet and then coordinate the update or recovery process. Dell describes BIOSConnect as follows:

“BIOSConnect provides a foundation platform allowing BIOS to connect to a Dell HTTPs backend and load an image via https method. This foundation expands the Serviceability feature set to enhance the on-box reliability experience by adding cloud-based Service OS (SOS) support.

BIOSConnect feature offers network-based SOS boot recovery capability by performing HTTP(s) download from the cloud to a local RAMDisk and transfers control to the downloaded Service OS image to perform the necessary corrective action. This enables the user to recover when the local HDD image is corrupted, replaced, or absent."

The details vary between the firmware update and OS recovery processes, but the high-level operation is the same — BIOSConnect connects to Dell's update infrastructure, and Dell delivers content needed to update or recover some of the most sensitive code on the device.


BIOSConnect Server Communication
1
Firmware Update or OS Recovery Request
Host Device
(BIOSConnect)
DELL
2
Content Delivery

Exploitation

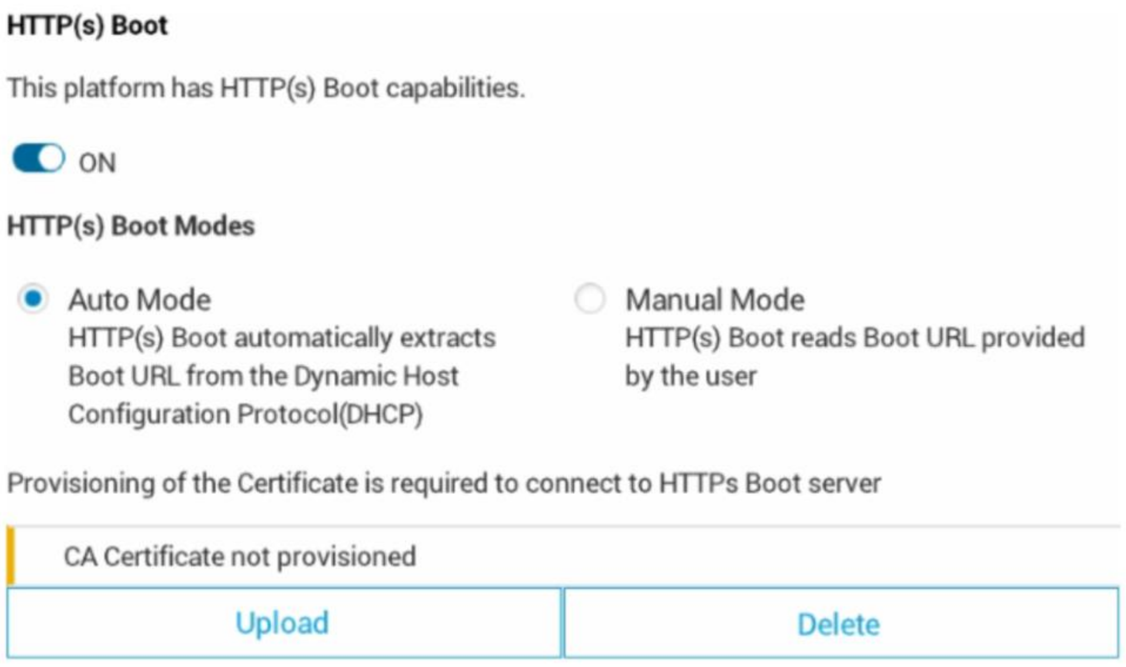
Eclypsium researchers have identified a series of four vulnerabilities that would enable a privileged network attacker to gain arbitrary code execution within the BIOS of vulnerable machines. The vulnerabilities were originally discovered on a Dell Secured-core PC Latitude 5310 using Secure Boot, and we later confirmed the issue on other models of desktops and laptops.

1. Insecure TLS Connection from BIOS to Dell – CVE-2021-21571. When attempting to connect to the backend Dell HTTP server, the TLS connection from BIOSConnect will accept any valid wildcard certificate. This allows an attacker with a privileged network position to impersonate Dell and deliver attacker-controlled content back to the victim device.

   The process of verifying the certificate for dell.com is done by first retrieving the DNS record from the hard-coded server 8.8.8.8 (Google) then establishing a connection to https://downloads.dell.com. However, any valid wildcard certificate issued by any of the built-in CA's contained within the BIOSConnect feature in BIOS will satisfy the secure connection condition, and BIOSConnect will proceed to retrieve the relevant files. The bundle of CA root certificates in the BIOS image was sourced from Mozilla's root certificate file (certdata.txt)

   When UEFI Secure Boot is disabled, this vulnerability can be used to gain arbitrary remote code execution in the UEFI/pre-boot environment on the client device without needing to take advantage of the additional buffer overflow vulnerabilities.

- Vulnerable HTTPS Boot configurations – Some HTTPS Boot configurations may also be exploitable due to using the same underlying verification code. When configuring HTTPS Boot, the user is required to provision a Certificate Authority (CA) certificate, which is intended to be used to verify connections to the remote boot server before allowing the HTTP Boot process to proceed. However, when this verification is performed, any valid certificate for any domain acquired from the same CA will be accepted, not just those for the configured remote boot server. Due to this limitation, CAs that issue certificates broadly (commercial, non-profit, cloud, web hosting etc.) should not be used for this role.



HTTP(s) Boot

This platform has HTTP(s) Boot capabilities.

ON

HTTP(s) Boot Modes

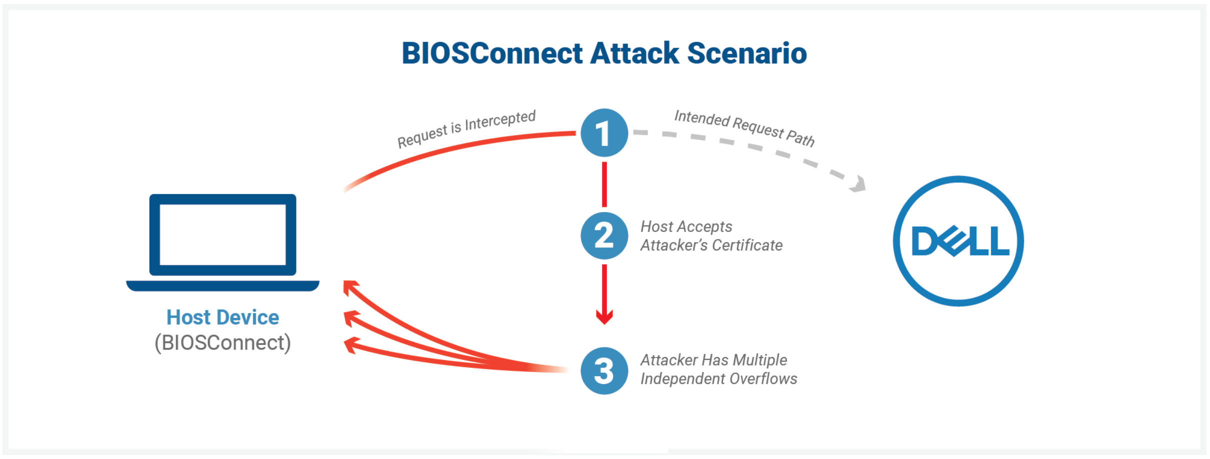
[✓ Auto Mode] HTTP(s) Boot automatically extracts Boot URL from the Dynamic Host Configuration Protocol(DHCP)

Manual Mode HTTP(s) Boot reads Boot URL provided by the user

Provisioning of the Certificate is required to connect to HTTPs Boot server

CA Certificate not provisioned

2. Overflow Vulnerabilities Enabling Arbitrary Code Execution – With the ability to impersonate Dell, the attacker can deliver malicious content back to the victim machine. Through subsequent analysis, we have identified three overflow vulnerabilities. Two of these vulnerabilities affect the OS recovery process, while the other affects the firmware update process. All three vulnerabilities are independent, and each one could lead to arbitrary code execution in BIOS. Eclypsium released additional detailed analyses of each vulnerability at DEFCON.


BIOSConnect Attack Scenario
Request is Intercepted
Intended Request Path
1
2
Host Accepts Attacker's Certificate
3
Attacker Has Multiple Independent Overflows
Host Device
(BIOSConnect)
DELL

As noted above, an attack scenario would require an attacker to be able to redirect the victim's traffic, such as via a Machine-in-the-Middle (MITM) attack. However, the virtually unlimited control over a device that this attack can provide makes it worth the effort by the attacker.

First, recent attacks have highlighted the great length that attackers will go to in order to compromise a vendor's supply chain and support infrastructure. Machine-in-the-Middle attacks are a relatively low bar to sophisticated attackers, with techniques such as ARP spoofing and DNS cache poisoning being well-known and easily automated. Additionally, enterprise VPNs and other network devices have become a top target of attackers, and flaws in these devices can allow attackers to redirect traffic. And finally, end-users working from home are increasingly reliant on SOHO networking gear. Vulnerabilities are quite common in these types of consumer-grade networking devices and have been exploited in widespread campaigns.

Successfully compromising the BIOS of a device would give an attacker a high degree of control over a device. The attacker could control the process of loading the host operating system and disable protections in order to remain undetected. This would allow an attacker to establish ongoing persistence while controlling the highest privileges on the device.

Impact

The problem affects 129 different models of Dell laptops, tablets, and desktops, and at least 30 million individual devices. The issue has been found on Secured-core PCs even if Secure Boot is enabled. You can refer further Dell's advisory for the full list of affected models

Mitigation

The system BIOS/UEFI will need to be updated for all affected systems. However, it's recommended that users not use BIOSConnect to perform this firmware update. Instead, it is advisable to run the BIOS update executable from the OS after manually checking the hashes against those published by Dell. According to Dell, all vulnerabilities have been remediated effective June 24, 2021.Dell has posted an advisory and provided the following recommended mitigations and workarounds

Vulnerability	Remediation
CVE-2021-21573, CVE-2021-21574	Remediated on the server side on May 28, 2021 and require no additional customer action.
CVE-2021-21571, CVE-2021-21572	Dell Client BIOS updates need to be applied to address the vulnerabilities. There are multiple ways to update your Dell Client BIOS: I.Dell notification solutions II.Drivers and Downloads III. Flashing the BIOS from the F12 One-Time Boot Menu

Dell recommends all customers update to the latest Dell Client BIOS version at the earliest opportunity. Customers who choose not to apply BIOS updates immediately or who are otherwise unable to do so at this time should apply the interim mitigation with disabling of the BIOSConnect and HTTPS Boot feature.

References

- https://eclypsium.com/2021/06/24/biosdisconnect/
- https://www.dell.com/support/kbdoc/000188682
- https://defcon.org/html/defcon-29/dc-29-speakers.html#shkatov

6.3 Buffer overflow

6.3.1 CVE-2021-3345 Libgcrypt version 1.9.0 has a heap-based buffer overflow

Introduction

Libgcrypt is a general-purpose cryptographic library originally based on code from GnuPG. It provides functions for all cryptographic building blocks: symmetric cipher algorithms (AES, Arcfour, Blowfish, Camellia, CAST5, ChaCha20 DES, GOST28147, Salsa20, SEED, Serpent, Twofish) and modes (ECB,CFB,CBC,OFB,CTR,CCM,GCM,OCB,POLY1305,AESWRAP), hash algorithms (MD2, MD4, MD5, GOST R 34.11, RIPE-MD160, SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE-128, SHAKE-256, TIGER-192, Whirlpool), MACs (HMAC for all hash algorithms, CMAC for all cipher algorithms, GMAC-AES, GMAC-CAMELLIA, GMAC-TWOFISH, GMAC-SERPENT, GMAC-SEED, Poly1305, Poly1305-AES, Poly1305-CAMELLIA, Poly1305-TWOFISH, Poly1305-SERPENT, Poly1305-SEED), public key algorithms (RSA, Elgamal, DSA, ECDSA, EdDSA, ECDH), large integer functions, random numbers and a lot of supporting functions.

Due to prior US export limitations on cryptographic software, the program is delivered through the European-based GnuPG server rather than the regular GNU archives. The most recent stable version, 1.10.1, was published on 2022-03-28.

A heap overflow is a type of buffer overflow that occurs when a piece of memory is allocated to the heap and data is written to it without any bound checking. This can result in the overwriting of essential heap data structures such as heap headers or any heap-based data such as dynamic object references, which can then result in the overwriting of the virtual function table.

While buffer overflow examples can be rather complex, it is possible to have very simple, yet still exploitable, heap-based buffer overflows:

```
#define BUFSIZE 256

int main(int argc, char **argv) {

  char *buf;

  buf = (char *)malloc(sizeof(char)*BUFSIZE);

  strcpy(buf, argv[1]);

}
```

CVSS Score - 7.8 / High

Description

The Libgcrypt group had hurried out a remedy for a severe problem in the free-source cryptography library's version 1.9.0. An exploit would allow an attacker to execute programs and write arbitrary data to a target system.

The security flaw is a heap-buffer overflow fault in Libgcrypt 1.9.0 (published on January 19; prior versions are unaffected), which researchers claimed may be exploited simply by decrypting a block of data.

According to Google Project Zero researcher Tavis Ormandy, who identified and disclosed the defect, the weakness is "easy to attack."

"Libgcrypt has a heap-buffer overflow owing to an improper assumption in the block buffer management function." "Simply decrypting any data might overflow a heap buffer with attacker-controlled data; no verification or signature is confirmed before the issue occurs," Ormandy noted in his research, which was published as part of Libgcrypt's alert.

Though the defective version is no longer accessible for download, it is unknown how many developers used it to construct their applications before it was removed. The authors of Libgcrypt recommend that developers replace the buggy library with the most recent version.

The weakness in the library affects Homebrew, according to cryptographer Filippo Valsorda. Homebrew is an open-source software package management system that streamlines software installation on Apple's macOS and Linux. The flaw was discovered and repaired by Homebrew's managers.

Exploitation

There is only one exploit that is publicly available and POCs for verifying the Libgcrypt < 1.9.1 Heap Buffer Overflow Vulnerability on Github:

- GitHub: https://github.com/MLGRadish/CVE-2021-3345

[CVE-2021-3345: POC exploit of CVE-2021-3345, a vulnerability in Libgcrypt version 1.9.0]

If you want to build a vulnerable Libgcrypt, follow these steps:

```
git clone --single-branch --branch LIBGCRYPT-1.9-BRANCH https://dev.gnupg.org/source/libgcrypt.git

cd libgcrypt

git checkout aa3f595341eb

./autogen.sh

./configure --enable-maintainer-mode && make
```

The use of exploit is mentioned clearly in the README.md file in the GitHub Repository.

Impact

Successful exploitation of this vulnerability may result in a complete compromise of the vulnerable systems.

Mitigation

Upgrade to Libgcrypt version 1.9.1 or later.

References

1. https://nvd.nist.gov/vuln/detail/CVE-2021-3345
2. https://threatpost.com/critical-libgcrypt-crypto-bug-arbitrary-code/163546/
3. https://stack.watch/product/gnupg/libgcrypt/

6.3.2 CVE-2021-23840 - Openssl Integer overflow in CipherUpdate

Introduction

OpenSSL is a general-purpose cryptographic library that implements the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols. SSL and TLS are widely used in network applications such as HTTP, IMAP, POP3, LDAP, and others to offer authentication, encryption, integrity, and non-repudiation services. X.509 certificates are used by clients and servers to communicate authentication information. The EVP functions (EVP_CipherUpdate, EVP_EncryptUpdate and EVP_DecryptUpdate) provide a high-level interface to OpenSSL cryptographic functions.

Integer overflow occurs when an integer (whole number) is attempted to be stored in computer memory that is too big for the integer data type in a specific system. For example, if an integer data type supports integers of up to two bytes or 16 bits in length (or an unsigned number of up to decimal 65,535) and two integers are put together that exceed the value of 65,535, the consequence is an integer overflow. Integer overflow can result, for example, in a request for dynamically allocated memory that is much too large or far too short for the program's needs. Integer overflows are frequently missed by the afflicted application.

The given vulnerability is triggered by the usage of externally supplied data for SIZE bytes for memory allocation, allowing a malicious user to damage the heap/memory.

CVSS Score - 7.5 / High

Description

Paul Kehrer noticed that OpenSSL treated some input lengths erroneously in EVP functions. Thus, it was triggered by the usage of externally supplied data for SIZE bytes for memory allocation, allowing a malicious user to damage the heap/memory.

A remote attacker might use this flaw to crash OpenSSL, resulting in a denial of service.

Applications may act erroneously or crash as a result of this vulnerability. This problem affects OpenSSL versions 1.1.1i and below.

Exploitation
Calls to EVP CipherUpdate, EVP EncryptUpdate, and EVP DecryptUpdate overrun the output length parameter if the input length is near to the platform's maximum allowable length for an integer. In such instances, the function call's return value will be 1 (indicating success), but the output length value will be negative.

This issue only affects OpenSSL-compiled programs that use the EVP CipherUpdate, EVP EncryptUpdate, or EVP DecryptUpdate methods. When these functions are supplied specifically designed values, the program may crash or act inappropriately.

Impact
An attacker might cause a buffer overflow, which would result in a core file and a denial of service (DoS).

Mitigation
Users of versions 1.1.1i and below should upgrade to the latest OpenSSL version (1.1.1n).

References
1. https://www.openssl.org/news/secadv/20210216.txt
2. https://nvd.nist.gov/vuln/detail/CVE-2021-23840

6.3.3 CVE-2021-3711 : SM2 Decryption Buffer Overflow

Introduction

According to the OpenSSL team's description, the data structure used to implement ASN.1 strings may allow an attacker to create a buffer overflow. This makes it possible to specifically crash the application as part of a denial of service attack by manipulating data packets.

CVSS Score - 9.8 / Critical

Description

In order to decrypt SM2 encrypted data an application is expected to call the API function EVP_PKEY_decrypt(). Typically an application will call this function twice.

The first time, on entry, the "out" parameter can be NULL and, on exit, the "outlen" parameter is populated with the buffer size required to hold the decrypted plaintext. The application can then allocate a sufficiently sized buffer and call EVP_PKEY_decrypt() again, but this time passing a non-NULL value for the "out" parameter.

A bug in the implementation of the SM2 decryption code means that the calculation of the buffer size required to hold the plaintext returned by the first call to EVP_PKEY_decrypt() can be smaller than the actual size required by the second call. This can lead to a buffer overflow when EVP_PKEY_decrypt() is called by the application a second time with a buffer that is too small. The location of the buffer is application dependent but is typically heap allocated.

Exploitation

A malicious attacker who is able to present SM2 content for decryption to an application could cause attacker chosen data to overflow the buffer by up to a maximum of 62 bytes altering the contents of other data held after the buffer, possibly changing application behaviour or causing the application to crash.

Impact

- OpenSSL versions 1.1.1k and below are affected by this issue.
- OpenSSL 1.0.2 is not impacted by this issue.
- OpenSSL 3.0 alpha/beta releases are also affected but this issue will be addressed before the final release.

Mitigation

This issue was reported to OpenSSL on 12th August 2021 by John Ouyang. The fix was developed by Matt Caswell in the OpenSSL 1.1.1l release. Users of versions 1.1.1k and below should upgrade to OpenSSL 1.1.1l.

References

1. https://www.openssl.org/news/secadv/20210824.txt

6.3.4 CVE-2021-43527 A remote code execution flaw was found in the way NSS verifies certificates

Introduction

Multiple NetApp products incorporate Libnss. NSS (Network Security Services) versions prior to 3.73 or 3.68.1 ESR are vulnerable to a heap overflow when handling DER-encoded DSA or RSA-PSS signatures.

CVSS Score - 9.8 / Critical

Description

Network Security Services (NSS) is Mozilla's widely used, cross-platform cryptography library

When you verify an ASN.1 encoded digital signature, NSS will create a VFYContext structure to store the necessary data. This includes things like the public key, the hash algorithm, and the signature itself.

```
struct VFYContextStr {
  SECOidTag hashAlg; /* the hash algorithm */
  SECKEYPublicKey *key;
  union {
     unsigned char buffer[1];
     unsigned char dsasig[DSA_MAX_SIGNATURE_LEN];
     unsigned char ecdsasig[2 * MAX_ECKEY_LEN];
     unsigned char rsasig[(RSA_MAX_MODULUS_BITS + 7) / 8];
  } u;
  unsigned int pkcs1RSADigestInfoLen;
  unsigned char *pkcs1RSADigestInfo;
  void *wincx;
  void *hashcx;
  const SECHashObject *hashobj;
  SECOidTag encAlg;    /* enc alg */
  PRBool hasSignature;

  SECItem *params;

};
```

Fig 1. The `VFYContext` structure from NSS

The maximum size signature that this structure can handle is whatever the largest union member is, in this case that's RSA at 2048 bytes. That's 16384 bits, large enough to accommodate signatures from even the most ridiculously oversized keys.

Okay, but what happens if you just....make a signature that's bigger than that?

Well, it turns out the answer is memory corruption. The untrusted signature is simply copied into this fixed-sized buffer, overwriting adjacent members with arbitrary attacker-controlled data.

The bug is simple to reproduce and affects multiple algorithms

This seems like a simple missing bounds check that should be easy to find. Coverity has been monitoring NSS since at least December 2008, and also appears to have failed to discover this.

Exploitation

Author been experimenting with alternative methods for measuring code coverage, to see if any have any practical use in fuzzing. The fuzzer that discovered this vulnerability used a combination of two approaches, stack coverage and object isolation.

Impact

Libnss versions prior to 3.73 or 3.68.1 are susceptible to a vulnerability which when successfully exploited could lead to disclosure of sensitive information, addition or modification of data, or Denial of Service (DoS). This vulnerability does NOT impact Mozilla Firefox. Applications using NSS for handling signatures encoded within CMS, S/MIME, PKCS #7, or PKCS #12 are likely to be impacted.

Email clients and PDF viewers that use NSS for signature verification, such as Thunderbird, LibreOffice, Evolution and Evince are believed to be impacted.

Mitigation

This vulnerability CVE-2021-43527 is resolved in NSS 3.68.1, 3.73.0. If you are a vendor that distributes NSS in your products, you will most likely need to update or backport the patch.

References

1. https://nvd.nist.gov/vuln/detail/CVE-2021-43527
2. https://support.f5.com/csp/article/K54450124
3. https://access.redhat.com/security/cve/cve-2021-43527

6.3.5 CVE-2021-3712 - Read buffer overruns processing ASN.1 strings

Introduction

According to the OpenSSL team's description, the data structure used to implement ASN.1 strings may allow an attacker to create a buffer overflow. This makes it possible to specifically crash the application as part of a denial of service attack by manipulating data packets.

CVSS Score - 7.4 / High

Description

ASN.1 strings are represented internally within OpenSSL as an ASN1_STRING structure which contains a buffer holding the string data and a field holding the buffer length. This contrasts with normal C strings which are repesented as a buffer for the string data which is terminated with a NUL (0) byte.

Although not a strict requirement, ASN.1 strings that are parsed using OpenSSL's own "d2i" functions (and other similar parsing functions) as well as any string whose value has been set with the ASN1_STRING_set() function will additionally NUL terminate the byte array in the ASN1_STRING structure.

However, it is possible for applications to directly construct valid ASN1_STRING structures which do not NUL terminate the byte array by directly setting the "data" and "length" fields in the ASN1_STRING array. This can also happen by using the ASN1_STRING_set0() function.

Numerous OpenSSL functions that print ASN.1 data have been found to assume that the ASN1_STRING byte array will be NUL terminated, even though this is not guaranteed for strings that have been directly constructed. Where an application requests an ASN.1 structure to be printed, and where that ASN.1 structure contains ASN1_STRINGs that have been directly constructed by the application without NUL terminating the "data" field, then a read buffer overrun can occur.

The same thing can also occur during name constraints processing of certificates (for example if a certificate has been directly constructed by the application instead of loading it via the OpenSSL parsing functions, and the certificate contains non NUL terminated ASN1_STRING structures). It can also occur in the X509_get1_email(), X509_REQ_get1_email() and X509_get1_ocsp() functions.

Exploitation

If a malicious actor can cause an application to directly construct an ASN1_STRING and then process it through one of the affected OpenSSL functions then this issue could be hit. This might result in a crash (causing a Denial of Service attack). It could also result in the disclosure of private memory contents (such as private keys, or sensitive plaintext).

Impact

- OpenSSL versions 1.1.1k and below are affected by this issue.
- OpenSSL versions 1.0.2y and below are affected by this issue. However OpenSSL 1.0.2 is out of support and no longer receiving public updates. Premium support customers of OpenSSL 1.0.2 should upgrade to 1.0.2za. Other users should upgrade to 1.1.1l.

Mitigation

An initial instance of this issue in the X509_aux_print() function was reported to OpenSSL on 18th July 2021 by Ingo Schwarze. The bugfix was developed by Ingo Schwarze and first publicly released in OpenBSD-current on 10th July 2021 and subsequently in OpenSSL on 20th July 2021 (commit d9d838ddc). Subsequent analysis by David Benjamin on 17th August 2021 identified more instances of the same bug. Additional analysis was performed by Matt Caswell. Fixes for the additional instances of this issue were developed by Matt Caswell. This issue was reported to OpenSSL on 12th August 2021 by John Ouyang. Users of the OpenSSL versions 1.1.1k and below should upgrade to OpenSSL 1.1.1l.

References

1. https://www.openssl.org/news/secadv/20210824.txt

6.4 Certifying Authority Best Practices gap

6.4.1 Google Chrome distrusts Spanish CA Camerfirma

Introduction

An entity that acts to verify the identities of entities (such as websites, email addresses, businesses, or individual people) and bind them to cryptographic keys through the issuance of electronic documents known as digital certificates is referred to as a certificate authority (CA).

Following are the functions done by CA Authority:-

1) Authentication is what a digital certificate offers by acting as a credential to confirm the legitimacy of the entity to which it is issued.
2) For safe communication over vulnerable networks like the Internet, encryption is used.
3) Integrity of papers signed with the certificate, preventing tampering during transmission by a third party.

CVSS Score - 6.4 / Medium

Description

In January 2021, Google Chrome announced that it will distrust the Certification Authority Camerfirma based in Spain from the upcoming version 90 of Google Chrome.There were 26 confirmed issues related to Camerfima. Some of them are listed below.

1) Online Certificate Status Protocol(OCSP) response did not comply with the standards.
2) One of Camerfirma's sub-CAs, Intesa Sanpaolo, also committed a number of errors in terms of prompt revocation. Even a certificate for "com.com" was issued due to "human mistake."
3) The key ID field's required for the issuer's name and serial number was broken by Camerfirma (you must not). Since 2003, all Camerfirma certificates have been performing this incorrectly. At the end of 2019, they said they had made a mistake and repaired it. However, they left the previously issued certificates in place. They did not revoke the certifications they reissued in 2020 that violated this policy.
4) For a variety of reasons, the world ceased believing in WoSign/StartCom as a CA in 2017. Camerfirma continued its partnership with StartCom in order to authenticate specific certificates, and it did so in accordance with the definition of "other techniques," which is the most peculiar (and last) approach which raises questions.
5) Many of their certificates had incorrect Subject Alternative Names fields, it was found. Since these reports "travelled to only one individual," who did not respond, they received no response from Camerfirma when they reported this. Even after cancelling some of these certificates, Camerfirma issued them wrongly again. Camerfirma never "deliberately" corrected certificates of this type.

Exploitation

This bug can be exploited by attacker by creating a website which looks similar to the original website and the user will not be able to differentiate between the legitimate and the malicious website, until he/she looks at the url, as both of them are not having the lock sign in the left corner. And the attacker can then spam the users with the fake website url and the users will get fooled which will impact the image of the organization.

When responding to CA issues in the past, different browser makers used a range of tactics, which were driven by both the circumstances of the occurrence and the technological alternatives available to the browsers. Google's decision to actively block all certificates, old and new, is based on the details available, the current technological implementation, and a desire to provide Chrome users and site administrators with a uniform, predictable, cross-platform experience.

Impact

When someone will visit the site having its Certificate from Camerfirma then they would not see the lock sign and the browser will show that the website is not trusted.

Please review the list of impacted root certificates below. It is worth noting that Google has supplied a comprehensive set of root certificates to maintain consistency across the numerous systems that Chrome supports, even if they are not intended for TLS usage. Please keep in mind that the limits apply only to the related subjectPublicKeyInfo fields of these certificates.

Affected Certificates (SHA-256 fingerprint)

- 04F1BEC36951BC1454A904CE32890C5DA3CDE1356B7900F6E62DFA2041EBAD51
- 063E4AFAC491DFD332F3089B8542E94617D893D7FE944E10A7937EE29D9693C0
- 0C258A12A5674AEF25F28BA7DCFAECEEA348E541E6F5CC4EE63B71B361606AC3
- C1D80CE474A51128B77E794A98AA2D62A0225DA3F419E5C7ED73DFBF660E7109
- 136335439334A7698016A0D324DE72284E079D7B5220BB8FBD747816EEBEBACA
- EF3CB417FC8EBF6F97876C9E4ECE39DE1EA5FE649141D1028B7D11C0B2298CED

Mitigation

Upgrade to versions above Google Chrome 90

Reference

1. https://www.feistyduck.com/bulletproof-tls-newsletter/issue_73_google_chrome_distrusts_camerfirma
2. https://www.theregister.com/2021/02/02/chrome_camerfirma_certificates/
3. https://business.blogthinkbig.com/26-reasons-chrome-does-not-trust-spanish-ca-camerfirma/
4. https://aboutssl.org/google-bans-another-misbehaving-ca-from-chrome/
5. https://groups.google.com/g/mozilla.dev.security.policy/c/dSeD3dgnpzk/m/iAUwcFioAQAJ

6.5 Other Categories

6.5.1 CVE-2021-23841 OpenSSL Null pointer deref in X509_issuer_and_serial_hash()

Introduction

The OpenSSL Project creates and maintains the OpenSSL software, which is a powerful, commercial-grade, feature-rich toolset for general-purpose encryption and secure communication. The OpenSSL Project is a collaborative effort to provide a strong, commercial-grade, full-featured, and Open Source toolkit that implements the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols, as well as a full-strength general-purpose cryptography library.

OpenSSL is licensed under an Apache-style license, which basically implies that you can obtain and use it for commercial and non-commercial uses as long as you follow a few simple licensing requirements.

When an application dereferences a pointer that it believes to be valid but is NULL, it often results in a crash or exit.

A null-pointer dereference occurs when a NULL-valued pointer is utilized as though it referred to a valid memory location.

While null-pointer dereferences are widespread, they are often easy to detect and remedy. They will always cause the process to crash until exception handling (on some platforms) is initiated, and even then, there is nothing that can be done to save the process.

CVSS Score - 5.9 / Medium

Description

The OpenSSL public API function X509 issuer and serial hash() tries to generate a unique hash value from the issuer and serial number data in an X509 certificate. However, it fails to handle any issues that may arise while processing the issuer field effectively (which might occur if the issuer field is maliciously constructed). This may result in a NULL pointer deref and a crash, perhaps leading to a denial of service attack. Because the method X509 issuer and serial hash() is never directly called by OpenSSL, applications are only susceptible if they utilize it directly and on certificates received from untrusted sources.

The software may dereference a null pointer, resulting in a NullPointerException. Null pointer errors are often caused by a breach of one or more programming assumptions. Most null pointer issues cause general software reliability issues, but if an attacker can intentionally cause a null pointer to dereference, the attacker may be able to use the resulting exception to bypass security logic or to cause the application to reveal debugging information useful in planning subsequent attacks.

Exploitation

Calls to EVP CipherUpdate, EVP EncryptUpdate, and EVP DecryptUpdate may overrun the output length parameter if the input length is near to the platform's maximum allowable length for an integer. In such instances, the function call's return value will be 1 (indicating success), but the output length value will be negative. Applications may act erroneously or crash as a result of this.

Because the method X509 issuer and serial hash() is never directly called by OpenSSL, applications are only susceptible if they utilize it directly and on certificates received from untrusted sources.

Impact

This vulnerability may subsequently result in a NULL pointer deref and a crash leading to a potential denial of service attack.

Mitigation

This problem affects OpenSSL versions 1.1.1i and below. These versions' users should update to OpenSSL 1.1.1j. This problem affects OpenSSL versions 1.0.2x and below. However, OpenSSL 1.0.2 is no longer supported and will no longer get public updates. Customers with premium support for OpenSSL 1.0.2 should update to 1.0.2y. Users need to update to 1.1.1j. OpenSSL 1.1.1j includes a fix (Affected 1.1.1-1.1.1i). OpenSSL 1.0.2y addresses this issue (Affected 1.0.2-1.0.2x).

Reference

1. https://www.openssl.org/news/secadv/20210216.txt
2. https://github.com/advisories/GHSA-84rm-qf37-fgc2
3. https://bugzilla.redhat.com/show_bug.cgi?id=1930310

6.5.2 Private Keys breached from Managed WordPress sites of GoDaddy

Introduction

A private key, also known as a secret key, is a variable in cryptography that is used with an algorithm to encrypt and decrypt data. Secret keys should only be shared with the key's generator or parties authorized to decrypt the data. Private keys play an important role in symmetric cryptography, asymmetric cryptography, and cryptocurrencies. Private keys share the following characteristics with passwords:

- They must be kept secret to be secure.
- They restrict access to data (private keys) or resources (passwords).
- Their strength depends on their length and randomness.

While passwords are usually limited to characters accessible from a computer keyboard, cryptographic keys can consist of any string of bits. Such strings may be rendered in human-accessible character sets, if necessary. Length and randomness are two important factors in securing private keys.

When a private key in a public-key infrastructure (PKI) environment is lost or stolen, compromised end-entity certificates can be used to impersonate a principal (a singular and identifiable logical or physical entity, person, machine, server, or device) that is associated with it. An end-entity certificate is one that does not have certification authority to authorize other certificates. Consequently, the scope of a compromise or loss of an end-entity private key is limited to only those certificates whose keys were lost.

The private key is the part of the certificate that the principal using PKI keeps private. In some cases, people think they may have lost control of private keys but are not sure. Since anyone in possession of a private key can impersonate its intended user, losing a private key is functionally equivalent to losing a password. A private key is harder to replace than a lost password, however, so losing a private key is harder to recover from.

CVSS Score - Critical

Description

An unknown attacker got unauthorized access to the infrastructure used to provide the company's Managed WordPress sites, affecting up to 1.2 million of its WordPress clients, according to GoDaddy.

According to the SEC (Securities and Exchange Commission) report, the attacker got access on September 6, 2021, using a compromised password, and was detected on November 17, 2021, when their access was canceled. While the business took a swift effort to repair the harm, the attacker had more than two months to build persistence, thus anyone utilizing GoDaddy's Managed WordPress package should presume compromise unless they can confirm otherwise.

GoDaddy acknowledged that they stored database passwords as plaintext or in a reversible format. These are also retrievable via their user interface. Unfortunately storing database passwords as plaintext is quite normal in a WordPress setting, where the database password is stored in the wp-config.php file as text. What is more surprising, in this breach, is that the password that provides read/write access to the entire filesystem via sFTP is stored as plaintext.

You might ask, what is sFTP?

The Secure File Transfer Protocol (SFTP) is a file transfer protocol that employs SSH encryption to safely transfer data between computers. It is included as part of SSH version 2.0. When transferring files, SFTP allows users to select the degree of authentication they want to employ. Users can transmit files using SFTP with no additional authentication, a user ID and password combination, or a set of SSH keys.

SFTP was created to be a more secure alternative to FTP (File Transfer Protocol). Despite apparent similarities, SFTP is a subsystem of SSH and distinct from FTP.

The following information is believed to have been accessed by the intruder -

- Email addresses and customer numbers of up to 1.2 million active and inactive Managed WordPress customers
- The original WordPress Admin password that was set at the time of provisioning was exposed
- sFTP and database usernames and passwords associated with its active customers, and
- SSL private keys for a subset of active customers

Exploitation

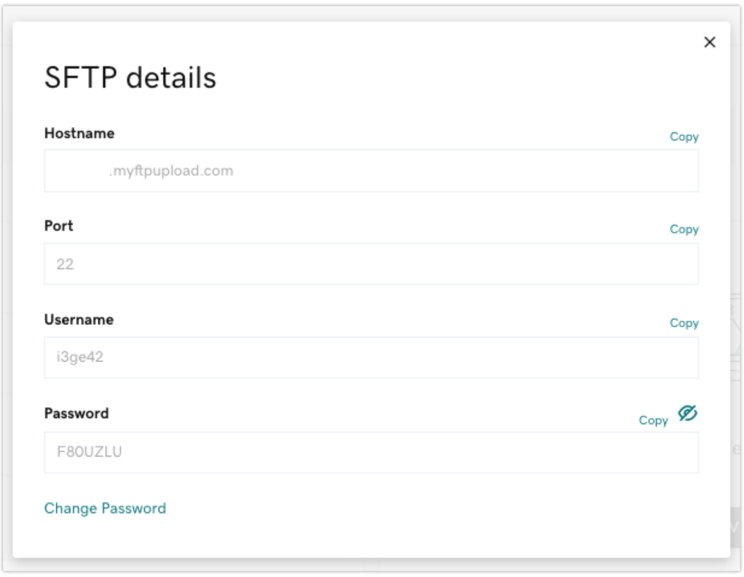
GoDaddy appeared to have stored sFTP credentials in plaintext or in a manner that might be reversed into plaintext. They chose this method over utilizing a salted hash or a public key, which are both regarded as industry best practices for sFTP. This gave an attacker instant access to password credentials without requiring them to be cracked.

According to the SEC filing, the attacker had access to user email addresses and customer numbers, as well as the original WordPress Admin password set at the time of provisioning and SSL private keys. All of these items may be useful to an attacker, but one in particular stuck out:

The attacker had access to active customers' sFTP and database usernames and passwords from September 6, 2021, to November 17, 2021.

Rather than maintaining salted hashes of these passwords or enabling public key authentication, GoDaddy kept sFTP credentials in such a way that the plaintext versions of the passwords could be accessed.

This was validated by entering the GoDaddy Managed Hosting user interface and seeing a personal password, as seen in the picture below. When employing public-key authentication or salted hashes, you cannot read your own password in this manner since the hosting provider does not hold it.


SFTP details

Hostname: .myftpupload.com

Port: 22

Username: i3ge42

Password: F80UZLU

Impact

While the SEC filing highlights the possible phishing risk provided by disclosed email addresses and customer numbers, this danger pales in comparison to the potential effect of leaked sFTP and database passwords.

Although GoDaddy promptly reset all of the vulnerable sites' sFTP and database passwords, the attacker had about a month and a half to take over these sites by uploading malware or installing a fraudulent administrator user. The attacker would be able to maintain persistence and control of the sites even after the passwords were reset.

Furthermore, given database access, the attacker would have had access to sensitive information, such as website customer PII (personally identifiable information) contained on the databases of the afflicted sites, and could have extracted the whole contents of all impacted databases. This includes password hashes contained in impacted sites' WordPress user accounts databases, as well as customer information from e-Commerce sites.

An attacker could get control of a site that had not changed its default admin password, but it would be easier for them to do so using their sFTP and database access.

An attacker who successfully performs a man-in-the-middle (MITM) attack that intercepts encrypted communication between a site visitor and an affected site might decode traffic using the stolen SSL private key on sites where the SSL private key was exposed.

Mitigation

GoDaddy reached out to the impacted customers over the next few days. Still given the severity of this issue and the data the attacker had access to, the following is recommended.

1. Change your WordPress passwords and, if feasible, compel a password reset for your WordPress users or customers.
2. Change any re-used passwords and encourage your users or customers to do the same.
3. Enable 2-factor authentication wherever possible.
4. Check your site for unauthorized administrator accounts.
5. Scan your site for malware using a security scanner.
6. Check the filesystem of your site, especially wp-content/plugins and wp-content/mu-plugins, for any odd plugins or plugins that do not display in the plugins menu, since legitimate plugins can be used to maintain unwanted access.
7. Keep an eye out for unusual emails - phishing is still a possibility, and an attacker might utilize extracted emails and customer numbers to get further sensitive information from victims of this intrusion.

References

1. https://www.bleepingcomputer.com/news/security/godaddy-data-breach-hits-12-million-managed-wordpress-customers/
2. https://trak.in/tags/business/2021/11/24/godaddy-hacked-12-lakh-users-critical-wordpress-data-exposed-what-to-do-next/
1. https://www.darkreading.com/attacks-breaches/godaddy-breach-exposes-ssl-keys-of-managed-wordpress-hosting-customers

6.5.3 Partitioning Oracle Attacks unveiled in USENIX conference Jan 2021

Introduction

Partitioning Oracle Attacks are similar to previous attacks considered in the password-authenticated key exchange (PAKE) literature. Password-Authenticated Key Exchange is a technique that allows several parties to establish shared cryptographic keys based on the same password knowledge. All parties can derive the same result from the password and the public messages they exchange. The unauthorized person who does not have the password cannot obtain it via the communication channel. There is no requirement for a trusted third party, such as public key infrastructure (PKI), to authenticate with the password.

PAKE prevents attacks from an eavesdropper or man-in-middle. The attackers are unable to guess the password without interaction with engaging parties. It can use weak human-memorable passwords to generate a high-entropy strong key to encrypt/decrypt, so it's very convenient in use and widely deployed in many commercial applications today.

CVSS Score - High

Description

A partitioning oracle arises when an adversary can

1. Efficiently craft ciphertexts that successfully decrypt under a large number of potential keys
2. Submit such ciphertexts to a system that reveals whether decryption under a target secret key succeeds. This enables learning information about the secret key.

The essential cryptanalytic stage in the assaults they experimented with was to build (what they termed) key multi-collisions, in which a single AEAD ciphertext may be constructed so that decryption works under some number k of keys. This cryptanalytic aim is formalized, and an algorithm for computing key multi-collisions for AES-GCM is provided. It generates key multi-collision ciphertexts of length O(k) in O(k^2) time utilizing polynomial interpolation from commercial libraries, making them scalable even to big k. An alternative polynomial interpolation approach allows for an algorithm that executes in time O(k log^2 k), although it is not accessible in standard library implementations to our knowledge. They provided more restricted attacks against XSalsa20/Poly1305 (as well as ChaCha20/Poly1305) and AES-GCM-SIV.

Given access to an oracle that reveals whether decryption succeeds, their key multi-collisions for AES-GCM enable a partitioning oracle attack that recovers the secret key in roughly m+logk queries in situations where possible keys fall in a set of size d = m· k. This will not work to recover much information about, e.g., random 128-bit keys where d = 2

128, but we show that it suffices to be damaging in settings where keys are derived from user-selected passwords or where key anonymity is important.

Exploitation

Two case studies were used to investigate partitioning oracles. First, they demonstrated how to construct a working partitioning oracle attack against Shadowsocks proxy servers. Shadowsocks was originally designed to aid with censorship avoidance in China, and it serves as the foundation for additional solutions such as Jigsaw's Outline VPN. Connections in Shadowsocks are protected via password-based AEAD with a user-chosen password exchanged between a client and the proxy server. They demonstrated how an attacker may use the proxy server as a partitioning oracle, despite the fact that it is supposed to discreetly dump faulty ciphertexts.

According to simulations based on password breach data, the attacker recovers the user's password 20% of the time by sending 124 ciphertexts to the server — several orders of magnitude fewer than the 60,000 required by a normal remote guessing assault. Because our attack ciphertexts are huge, the latter requires less overall bandwidth, but to succeed 70% of the time, our attack requires fewer queries and less overall bandwidth than the remote guessing attack. They revealed our exploits to the Shadowsocks community appropriately and assisted them in mitigating the issue. Then they moved on to password-authenticated key exchange (PAKE). In this section, we focus on improper implementations of the OPAQUE protocol, which was recently recognized as a candidate for standardization by the IETF's Crypto Forum Research Group (CFRG).

Impact

In a nutshell, this is a faster password search on non-committing AEAD schemes such as AES-GCM and ChaCha20-Poly1305 with few oracle calls. You are protected if your password is strong enough.

A committing encryption scheme is a scheme that computationally intractable to find a pair of keys and a ciphertext that decrypts under both keys.

The defined targeted multi-key collision resistance (TMKCR) security by the following game in which for a given target key set $K \in \mathcal{K}$ for a randomized adversary A must produce a nonce $N*$, associated data $AD*$, and a ciphertext $C*$ such that $\text{AuthDec}K(N*,AD*,C*) \neq \perp$ for all $K \in \mathcal{K}$. And the advantage is defined as

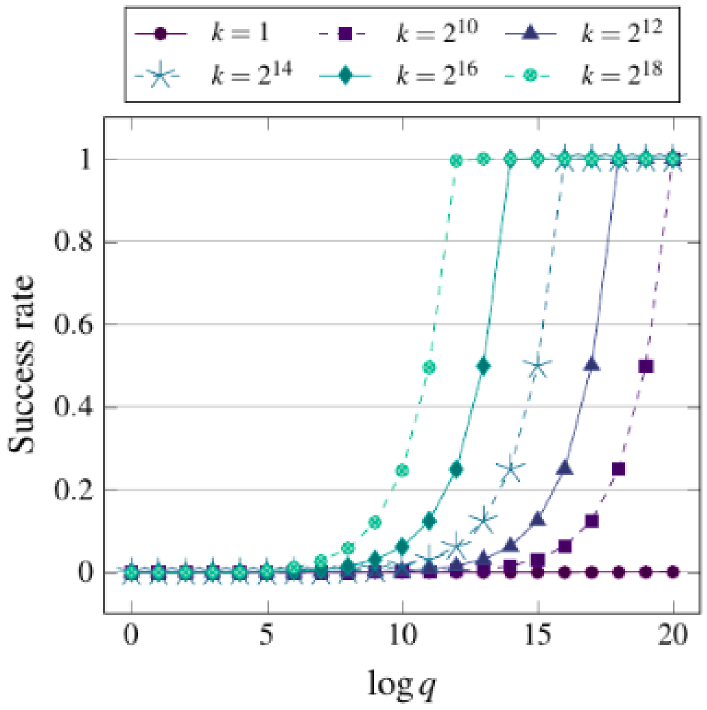
$$\mathbf{Adv}^{tmk-cr}_{\mathrm{AEAD},\mathbb{K}} = \Pr\left[\mathsf{TMKCR}^{\mathcal{A}}_{\mathrm{AEAD},\mathbb{K}}\right] \implies \textsf{true}$$

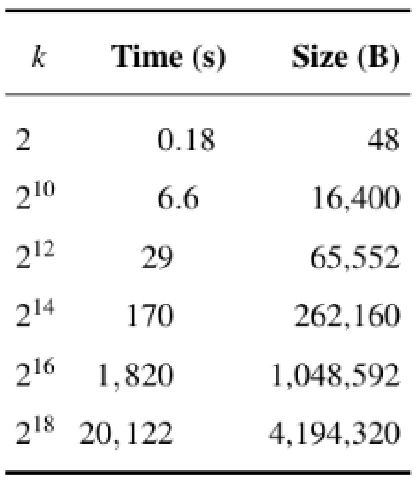
As one can see, if the attacker somehow partitions the keyspace then they can reduce the oracle queries this means the attack has less online interaction to be detected.

For example, consider AES-GCM which is not committing, and select two keys and an arbitrary ciphertext then the adversary will look for creating a valid tag so that when during decryption the tag is valid (note; AES-GCM doesn't require decryption to validate the authentication tag, however, separating decryption and tag validation requires to pass on the data, and usually, this is not preferred for performance reasons)

The aim is the recovery of a password pw. Consider that you want to test the membership of two passwords S∗1={pw1,pw2}. Create two keys K1=PBKDF(salt,pw1) and K2=PBKDF(salt,pw2) (the salt can be found by sniffing!), now use Dodis et, al's approach [1], construct a ciphertext C′ with a tag such that it decrypts correctly under the keys K1 and K2. Now send the splitting value V^ to the server. If the server indicates the decryption is successful then pw∈S∗1. With iterating this procedure the attacker can find the password in |D|/2+1 queries while the default brute-force requires |D| queries. This attack has degree 2, if k degree is possible then this will decrease the query amount substantially.

With SageMath code they were able to find k=216 colliding ciphertext very quickly. They simulate their adaptive partitioning oracle attack that constructs splitting ciphertexts of size k iteratively for different sets of keys until the oracle returns valid. At this point, the adversary performs a binary search in logk queries to find the secret.





k	Time (s)	Size (B)
2	0.18	48
2^{10}	6.6	16,400
2^{12}	29	65,552
2^{14}	170	262,160
2^{16}	1,820	1,048,592
2^{18}	20,122	4,194,320

On the left, on the above picture, their success rate, and on the right is the time to generate the k-way collisions.

It turns out that a multi-key collision attack on the Poly1305 is harder than GCM due to the appended 0x01. They are only able to generate at most 10, compare to GCM which reaches 18 degrees. this still gives a factor- of-ten speedup in partitioning oracle attacks

Misuse-resistant schemes like AES-GCM-SIV are also applicable to this attack.

Mitigation

- Limiting the length of the ciphertext
- Entropy requirements on shared secrets
- Use committing AEAD scheme, there are non in any standards, yet!
- Use single key Encrypt-then-HMAC (long live HMAC) or KMAC.

References

1. https://eprint.iacr.org/2020/1491.pdf
2. https://www.usenix.org/conference/usenixsecurity21/presentation/len
3. https://www.youtube.com/watch?v=h-T1bQTt4_Y

6.5.4 CVE-2021-20232 – vulnerability in gnutls may lead to memory corruption

Introduction

GnuTLS is a free software implementation of the TLS, SLL and DTLS protocols. It was discovered that GnuTLS incorrectly handled certain memory operations. A remote attacker could possibly use this issue to cause GnuTLS to execute arbitrary code , crash the application or program. CVE-2021-20232 is the CVE depicting the vulnerability related to gnuTLS improper handling of memory operations and the impacts caused by the same.

CVSS Score - 9.8 / Critical

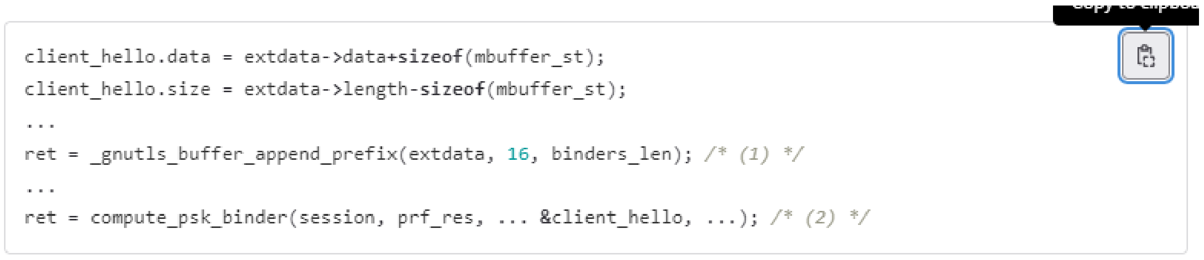
Description

GnuTLS is a free software implementation of the TLS, SSL and DTLS protocols. A flaw was found in gnutls. A use after free issue in client_send_params in lib/ext/pre_shared_key.c may lead to memory corruption and other potential consequences.

This issue affects the function client_send_params in the library *lib/ext/pre_shared_key.c*. The manipulation with an unknown input lead to a memory corruption vulnerability. Referencing memory after it has been freed can cause a program to crash, use unexpected values, or execute code. Impacted is confidentiality, integrity, and availability.

Exploitation

1. Client sending a “key_share” or “pre_share_key” extension may result in the dereferencing the pointer no longer valid after realloc().
2. A remote attacker can trick the victim to connect to a malicious server using a large Client Hello message over TLS 1.3, trigger a use-after-free error and crash the application or execute arbitrary code on the system.



```
client_hello.data = extdata->data+sizeof(mbuffer_st);
client_hello.size = extdata->length-sizeof(mbuffer_st);
...
ret = _gnutls_buffer_append_prefix(extdata, 16, binders_len); /* (1) */
...
ret = compute_psk_binder(session, prf_res, ... &client_hello, ...); /* (2) */
```

At (1) realloc may be called on extdata->data, and that results in client_hello.data being no longer valid at (2).

Impact

Any client component utilizing GnuTls is impacted to arbitrary code execution , program crash or code execution scenarios. This depends on the sensitivity of the client data it is processing and hence, as it is affecting confidentiality, integrity and availability aspects of client data, this is critical level vulnerability.

Mitigation

Upgrade Alpine:3.10 gnutls to version 3.6.15-r1 or higher.

References

1. https://nvd.nist.gov/vuln/detail/CVE-2021-20232
2. https://bugzilla.redhat.com/show_bug.cgi?id=1922275

6.5.5 A Security Analysis of STARTTLS in the Email Context

Introduction

The security analysis of STARTTLS discovered that there are many security vulnerabilities in the STARTTLS usage especially using SMTP and IMAP protocols while sending or consuming email requests or responses. The detailed level of security analysis, impact and mitigation strategies are described in below sections.

CVSS Score - 5.3 / Medium

Description

TLS is one of the best analysed and most widely used encryption technologies.

But for historical reasons, TLS for email protocol Is generally not used directly but negotiated through STARTTLS. However, it added complexity and became liable to security vulnerabilities, such as command injection attacks and naive STARTTLS stripping.

Security analysis of STARTTLS exposed many different vulnerabilities linked with an opportunistic encryption method in the email clients and servers. The abuse of STARTTLS vulnerabilities could open the gateway to the intended man-in-the-middle attacks, allowing an invader to produce mailbox content and steal user credentials.

It is a fact that STARTTLS is vulnerable and should be avoided due to being under-specified in the standards. However, there are still hundreds of thousands of email servers and millions of email clients that support STARTTLS.

Exploitation

The below is the analysis list of vulnerabilities exploitation scenarios because of STARTTLS weaknesses:

1. Stripping attack: When the Meddler in the middle (MITM) removes the capability of STARTTLS to server response, it can be easily downgraded to plaintext communication resulted in the confidentiality attacks.
2. Command Injection: In the postfix SMTP server, it was found a command injection is possible in the Postfix SMTP server. Whenever a client appends an extra COMMAND after the STARTTLS command, that command is buffered and evaluated after the transition to TLS.

In effect, this allows an attacker to inject plaintext command into the encrypted session.

#3. PREAUTH Command

An attacker bypassed the STARTTLS by sending the PREAUTH command. It's easy to see as the client proceeded to SELECT the inbox instead of terminating the connection. At this point, the intruder has full control over the client and only needs to mimic an IMAP server for tampering with the client's mailbox data. Sensitive data is leaked if the client synchronizes the draft and the sent emails. However, as PREAUTH notifies the MUA that is already authenticated, it will not expose user credentials.

#4. Malicious Redirects

Mailbox referrals are helpful when combined with PREAUTH greetings. As an attacker bypass STARTTLS security with a PREAUTH greeting, they can escalate the problem by responding with a redirect to the client's SELECT command. It notifies the client that the selected mailbox is available on another server. The attacker can select a domain and use a server for which they have a valid X.509 certificate. It will immediately leak the user credential to the hacker if the client uses this referral.

Impact

Credential Theft :

The Stripping attack was used to steal user credentials using IMAP and SMTP commands. Using meddler in the middle attack, the plaintext commands are possible to inject that causes the server to execute them that affects credential theft .

IMAP Connection Downgrade

In an IMAP protocol, the server can alert the client in the first mail that it has been authenticated by the PREAUTH command. The protocol prohibits the use of the STARTTLS command in the authenticated state. Thus, if a client app accepts the PREAUTH command, it can't enforce STARTTLS. A Meddler-in-the-middle attacker may use it to prevent the STARTTLS to upgrade the connection and force a client to use an unencrypted connection.

This issue is severe in combination with the IMAP features Mailbox Referrals and Login Referrals. These commands let a server instruct a client to log into another IMAP server. An attacker can use these referrals to force clients to send credentials using the PREAUTH to prevent an encrypted connection.

Mitigation

1. Recommendations For Email Client Users

Email clients should authenticate themselves with a username and strong password before sending a new email or opening existing emails. Moreover, the transitions towards TLS using STARTTLS must be enforced because a downgrade can reveal the username and password and allow an intruder to access the email account.

According to the researchers, implicit TLS is more secure than STARTTLS, and for users, it is recommended to configure their email clients to use POP3, SMTP, and IMAP with implicit TLS on dedicated ports, such as SMTP submission on port 465, IMAP on port 993, and POP3 on port 995.

2. Recommendations For Mail Server Administrators

Make sure to use secure connections while securing your mail servers. Encrypt IMAP and POP3 authentication and use TLS and SSL. Moreover, limit the number of connection and authentication errors to avoid attacks. Eliminate unused functionalities from the server by disabling unnecessary default settings. Mail server administrators should avoid being an open relay for intruders by specifying the domains and IP addresses.

To protect your mail servers from unauthorized access, implement access control and authentication. For example, SMTP requires users to have a username and password to send mail from the server. Check DNS-based checklist and do not accept emails from any IPs or domains listed on them.

3. Recommendation For Application Developers

Both the client applications and email servers should provide implicit TLS by default. Software developers may not support STARTTLS in the long term and thus simplify their code and configuration files and dialogs. It is recommended to audit all applications supporting STARTTLS, both on the server and client-side, to discover bugs. Moreover, applications need to ensure that no unencrypted content is processed as part of an encrypted connection. PREAUTH in combination with the STARTTLS must not be allowed by the IMAP application.

References

1. https://thesecmaster.com/what-is-starttls-how-starttls-vulnerabilities-affect-popular-email-clients/

6.5.6 Arbitrary Signature Forgery in Stark Bank ECDSA Libraries disclosed with Multiple CVEs (CVE-2021-43572, CVE-2021-43570, CVE-2021-43569, CVE-2021-43568, CVE-2021-43571)

Introduction

Stark Bank is a financial technology company that provides services to simplify and automate digital banking, by providing APIs to perform operations such as payments and transfers. In addition, Stark Bank maintains a number of cryptographic libraries to perform cryptographic signing and verification.

CVSS Score - 9.8 / Critical

Description

Stark Bank cryptographic libraries are meant to be used to integrate with the Stark Bank ecosystem, but are also accessible on popular package manager platforms in order to be used by other projects. The node package manager reports around 16k weekly downloads for the ecdsa-node implementation while the Python implementation boasts over 7.3M downloads in the last 90 days on PyPI. A number of these libraries suffer from a vulnerability in the signature verification functions, allowing attackers to forge signatures for arbitrary messages which successfully verify with any public key.

Exploitation

The (slightly simplified) ECDSA verification of a signature (r, s) on a hashed message z with public key Q and curve order n works as follows:

- Check that r and s are integers in the $[1, n-1]$ range, return *Invalid* if not.
- Compute $u_1 = zs^{-1} \mod n$ and $u_2 = rs^{-1} \mod n$.
- Compute the elliptic curve point $(x, y) = u_1G + u_2Q$, return *Invalid* if (x, y) is the point at infinity.
- Return *Valid* if $r \equiv x \mod n$, *Invalid* otherwise.

The ECDSA signature verification functions in the libraries listed above fail to perform the first check, ensuring that the r and s components of the signatures are in the correct range. Specifically, the libraries are not checking that the components of the signature are non-zero, which is an important check mandated by the standard, see X9.62:2005, Section 7.4.1/a:

1. If r’ is not an integer in the interval [1, n-1], then reject the signature.

2. If s’ is not an integer in the interval [1, n-1], then reject the signature.

For example, consider the following excerpt of the verify function from the ecdsa-python implementation.

```
def verify(cls, message, signature, publicKey, hashfunc=sha256):
    byteMessage = hashfunc(toBytes(message)).digest()
    numberMessage = numberFromByteString(byteMessage)
    curve = publicKey.curve
    r = signature.r
    s = signature.s
    inv = Math.inv(s, curve.N)
    u1 = Math.multiply(curve.G, n=(numberMessage * inv) % curve.N, N=curve.N, A=curve.A,
P=curve.P)
    u2 = Math.multiply(publicKey.point, n=(r * inv) % curve.N, N=curve.N, A=curve.A, P=curve.P)
    add = Math.add(u1, u2, A=curve.A, P=curve.P)
    modX = add.x % curve.N
    return r == modX
```

In that code snippet, the values r and s are extracted from the signature without any range check. An attacker supplying a signature equal to (r, s) = (0, 0) will not see their signature rejected. Proceeding with the verification, this function computes the inverse of the s component. Note that the Math.inv() function returns zero when supplied with a zero input (even though 0 does not admit an inverse). The code then computes the values u1 = inv * numberMessage * G and u2 = inv * r * Q, but since inv is zero, u1 and u2 will both be zero, i.e., the point at infinity, regardless of the value of numberMessage (the message hash, which we called z above) and Q (the public key). Subsequently, the implementation computes the intermediary curve point add by adding up the two previously computed points, which again results in the point at infinity. The final line checks that the r-component of the signature is equal to the x-coordinate of the curve point, essentially checking that 0 == 0 for all any message and any public key. Therefore, a signature (r, s) = (0, 0) is deemed valid by the code for any message, and under any public key.

Impact

An attacker can forge signatures on arbitrary messages that will verify for any public key. This may allow attackers to authenticate as any user within the Stark Bank platform, and bypass signature verification needed to perform operations on the platform, such as send payments and transfer funds. Additionally, the ability for attackers to forge signatures may impact other users and projects using these libraries in different and unforeseen ways.

Mitigation

Users of the different Stark Bank ECDSA libraries should update to the latest versions. Specifically, versions larger or at least equal to the following should be used.

- ecdsa-python: v2.0.1
- ecdsa-java: v1.0.1
- ecdsa-dotnet: v1.3.2
- ecdsa-elixir v1.0.1
- ecdsa-node v1.1.3

References

1. Arbitrary Signature Forgery in Stark Bank ECDSA Libraries disclosed with Multiple CVEs

6.5.7 CVE-2021-41117 Insecure Random Number Generation found in keypair

Introduction

This issue was reported to GitHub Security Lab by Ross Wheeler of Axosoft. It was discovered by Axosoft engineer Dan Suceava, who noticed that keypair was regularly generating duplicate RSA keys. GitHub security engineer @vcsjones (Kevin Jones) independently investigated the problem and identified the cause and source code location of the bug.

CVSS Score - 9.1 / Critical

Description

keypair implements a lot of cryptographic primitives on its own or by borrowing from other libraries where possible, including node-forge. An issue was discovered where this library was generating identical RSA keys used in SSH. This would mean that the library is generating identical P, Q (and thus N) values which, in practical terms, is impossible with RSA-2048 keys. Generating identical values, repeatedly, usually indicates an issue with poor random number generation, or, poor handling of CSPRNG output.

Exploitation

Issue 1: Poor random number generation (GHSL-2021-1012)

The library does not rely entirely on a platform provided CSPRNG, rather, it uses it's own counter-based CMAC approach. Where things go wrong is seeding the CMAC implementation with "true" random data in the function defaultSeedFile. In order to seed the AES-CMAC generator, the library will take two different approaches depending on the JavaScript execution environment. In a browser, the library will use window.crypto.getRandomValues(). However, in a nodeJS execution environment, the window object is not defined, so it goes down a much less secure solution, also of which has a bug in it.

It does look like the library tries to use node's CSPRNG when possible: https://github.com/juliangruber/keypair/blob/87c62f255baa12c1ec4f98a91600f82af80be6db/index.js#L1016

Unfortunately, it looks like crypto is null because a variable was declared with the same name, and set to null: https://github.com/juliangruber/keypair/blob/87c62f255baa12c1ec4f98a91600f82af80be6db/index.js#L759

So the node path is never taken.

However, when window.crypto.getRandomValues() is not available, a Lehmer LCG random number generator is used to seed the CMAC counter, and the LCG is seeded with

Math.random. While this is poor and would likely qualify in a security bug in itself, it does not explain the extreme frequency in which duplicate keys occur.

Main flaw

The output from the Lehmer LCG is encoded incorrectly. The specific line with the flaw is:

```
b.putByte(String.fromCharCode(next & 0xFF))
```

The definition of putByte is

```
util.ByteBuffer.prototype.putByte = function(b) {
 this.data += String.fromCharCode(b);
};
```

Simplified, this is String.fromCharCode(String.fromCharCode(next & 0xFF)). The double String.fromCharCode is almost certainly unintentional and the source of weak seeding. Unfortunately, this does not result in an error. Rather, it results most of the buffer containing zeros. An example generated buffer:

(Note: truncated for brevity)

```
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x04\x00\x00\x00....\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
```

Since we are masking with 0xFF, we can determine that 97% of the output from the LCG are converted to zeros. The only outputs that result in meaningful values are outputs 48 through 57, inclusive.

The impact is that each byte in the RNG seed has a 97% chance of being 0 due to incorrect conversion. When it is not, the bytes are 0 through 9.

In summary, there are three immediate concerns:

1. The library has an insecure random number fallback path. Ideally the library would require a strong CSPRNG instead of attempting to use a LCG and Math.random.
2. The library does not correctly use a strong random number generator when run in NodeJS, even though a strong CSPRNG is available.
3. The fallback path has an issue in the implementation where a majority of the seed data is going to effectively be zero.

Impact

Due to the poor random number generation, keypair generates RSA keys that are relatively easy to guess. This could enable an attacker to decrypt confidential messages or gain authorized access to an account belonging to the victim. A bug in the pseudo-random number generator used by keypair versions up to and including 1.0.3 could allow for weak RSA key generation.

Mitigation

The bug in the pseudo-random number generator is fixed in commit 9596418.

- If the crypto module is available, it is used instead of the pseudo-random number generator.

References

1. https://github.com/juliangruber/keypair/security/advisories/GHSA-3f99-hvg4-qjwj
2. https://github.com/juliangruber/keypair/commit/9596418d3363d3e757676c0b6a8f2d35a9d1cb18
3. https://github.com/juliangruber/keypair/blob/87c62f255baa12c1ec4f98a91600f82af80be6db/index.js#L1008