CVE Analysis: CVE-2020-0601

Short description:

A certificate usually includes a subject, date of expiration, certificate main task (key usage), a public key number, and an Algorithm (in this case, an Elliptic Curve). There are many curves to be used, each curve with different parameters. When first encountering a certificate Windows checks the source of the certificate, and then saves the certificate public key into it's "Certificate Cache". However, when accessing this certificate again for the second time, the public key parameter is the only thing being compared to in this "Certificate Cache" library. Therefore, a malicious key could be created with the same public key, but with a different Algorithm/Curve parameter. Having a validated ECC (Elliptic Curve Certificate) certificate can be detrimental because it is a method of transmitting encrypted data (possibly by means of TLS(Transport Layer Security)). With a man in the middle attack, they could send in their own data to display, or send in a bug to retrieve data.

Who can launch the attack?:

To generate the correct public key Q, one can brute force Q = x * G, where G is the algorithm/curve parameter and x is the hacker's private key, by changing G randomly to receive a correct Q, the public key. The easiest solution is to set the private key to 1 and set the known public key equal to the Generator. Then you submit this certificate with the variables 1 and G, with a spoofing software (from where this public key was originally associated with) and it will always equal to Q. Anyone with the server port and the correct spoofing location to a website is able to infiltrate the system with this CVE. All you need to have is the correct Public Key generated from the equation. Cryptographers can find the smallest of attacks and link it to a great span of attacks using this CVE.

Why does the attack matter?:

The attack matters because anything using Microsoft CryptoAPI is affected. Microsoft has already patched this glitch by tracking if the certificate was tampered. However, any type of

malicious code could be sent to a system without the patch to perform a variety of tasks, ranging from MITM, DNS Tunneling, and Malware. Anyone running this system will be affected and the attacker can do almost anything they can perceive of. Industries like healthcare, bank systems, social media, and many more fields of applications with personal information would have been vulnerable if the patch was not installed.

What damage does the CVE cause?:

Because this vulnerability would result in incredibly widespread data breach, users would have to disconnect everything from devices currently using the Microsoft CryptoAPI from the network. Remote Code Execution without authority is a possible attack an attacker could pull off if their malicious certificates got verified. One could also send in a certificate to get data from the place of breach and redirect it to cause a privacy leak. It's like the chosen plaintext attack where the attacker finds the key using before and after images to search and decrypt messages.

What is the attack?:

An attacker would exploit this vulnerability to create their own cryptographic certificates, which would appear to originate from a legitimate certificate that is fully trusted by Windows by default.

The attacker could let you download/install malware that pretended to be a legit software update.

Examples of validation of trust may be impacted include:

- HTTPS connections
- Signed files and emails
- Signed executable code launched as user-mode processes

How does it work?:

To launch an attack:

1. Create a spoofing CA. Which is created with the same public key and parameters of a trusted CA. Set the generator to a value, where you know the private key.
2. Create a certificate signing request.
3. Sign the created certificate signing request with your spoofed CA and CA key.
4. Bundle the signed certificate request with the spoofed CA.

When Windows checks this certificate and sees it has been signed, it will then look at the public key of the created CA. It will see that the key matches with the key of a trusted CA, and it will verify the signature of the spoofed CA with the spoofed CA's generator.

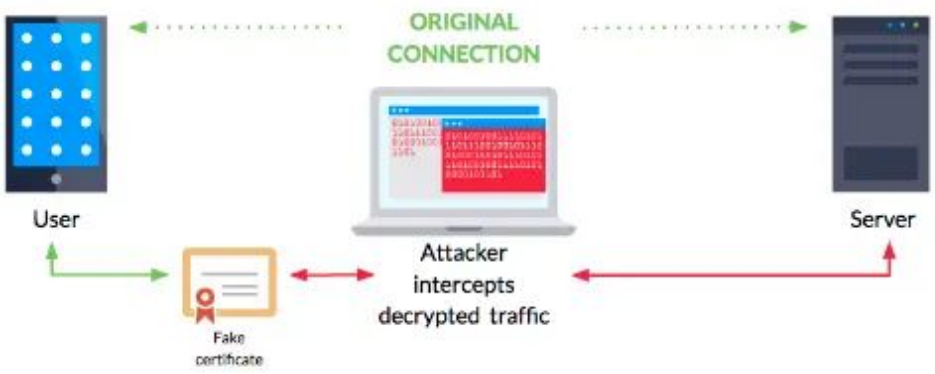
How was the bug fixed?:

The vulnerability exists only on Windows 10 and Windows Server 2016 and 2019. Adding on to this, other Windows OS do not support ECC certificates. Firefox on the other hand does support ECC certificates, but has never supported Crypt32.dll, another name for Microsoft's CryptoAPI. Firefox uses NSS (Network Security Services) for HTTPS connections. The company Microsoft has added a stopgap by implementing a check for all certificates to see if they were tampered with. If the certificate was tampered with, it is logged with Microsoft's logging mechanism and blocked. It is a real fix unless someone can figure out a way to disable the certificate checker. Along with the already limited OS that this vulnerability could be run on, the vulnerability was discovered early by NSA's research team and no known attacks were published. Given this information, NSA has warned of the uprising of more spoofing applications after the CVE-2020-0601 information release. ECC certificates are a way to transfer data over TLS/SSL.

Here is a website which shows where ECC certificates are supported on Browsers:
https://www.tbs-certificates.co.uk/navigateursECC.html.en

Here is a website which shows where ECC certificates are supported on Operating Systems:
https://support.globalsign.com/ssl/ssl-certificates-life-cycle/ecc-compatibility

Threat model:



Improper certificate validation

Source:https://learnworthy.net/wp-content/uploads/2020/02/Screenshot-2020-02-18-at-11.19.39.png

Assume there is a company Cryptex that validates public encryption keys using CryptoAPI. The keys used for this algorithm has two parts: the bytes that help define the encryption key (the public key itself), and the actual metadata that is provided as parameters to the algorithm. Cryptex, unfortunately, is a bit lazy with how they validate these public encryption keys. Instead of checking both the bytes and the parameters in order to confirm that the certificate is legitimate, Cryptex only checks the bytes. Suppose there was an attacker Bob who wanted to spoof a certificate and have it be trusted by Cryptex so that he could lure victims to malicious endpoints and perform man-in-the-middle attacks. Because Cryptex only checks the encryption key, Bob could have his own certificate validated by using a curve generator that, when combined with his own private key, yields the public key that will be checked by Cryptex. Because the public key P is defined as x * G, where x is the private key and G is the generator, the easiest way for Bob to do this would be to simply have x equal 1. In this case the equation would simplify to P = G, so Bob simply has to make his Generator equal the Public key and his malicious certificate will pass Cryptex's check.

Cryptex, none the wiser, sees that the key on Bob's malicious certificate matches with the key stored in their Trusted Certificates cache, so it allows his certificate to go through. Now that his certificate has been accepted, Bob can perform man-in-the-middle attacks on victims visiting sites through his certificate and gain access to information that should have been confidential.

Vector justification:

Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N

Attack vector: Network

The assessment for the Attack vector is accurate. This vulnerability can be exploited remotely over the network layer. Since CryptoAPI doesn't check for ECC used from certification authority, the attacker can simply spoof a malicious certification authority server with a public key value that is stored in Windows' certification cache. Doing so gives the attacker the ability to inject malicious executable code to the target devices which can lead to huge security issues.

Attack complexity: Low

We agree with this assessment. Although ECC as a cryptographic algorithm can be really hard to break, given the ability to modify the generator and private key, it is still pretty easy to generate a certain public key. Namely, there's no specialized access conditions. One example is for the generator to have the same value as the target public key and private key have the value of 1.

Privileges required: None

This assessment is accurate in our understanding. The attacker doesn't require any form of authorization prior to any attack. As long as the attacker has access to at least one public key that is authorized by the target, they would be able to initiate an attack.

User interaction: Required

User interaction is obviously required for this vulnerability to be exploited. The prerequisite for the attack to happen is that there's some public key stored in the system certification cache. This would require users to interact with networks and authenticate some authority. Besides, after the attacker set up malicious authority, it would still require user interaction for the attacker to communicate with their target.

Scope: Unchanged

We think the scope is unchanged. The intention of CryptoAPI is for Windows devices to validate ECC certificates. Due to the vulnerability of CryptoAPI, those Windows devices would validate unauthorized malicious certificates. The scope indeed remains unchanged for this vulnerability.

Confidentiality impact: High

It is clear that the confidentiality of the impacted component is badly compromised. Once the attacker successfully injects executable code into the target, they would have all privilege and access to the target device.

Integrity impact: High

The attacker would get full access to all privileges and all files of the Windows device protected by CryptoAPI once they successfully deploy the attack. Therefore, it is a total loss of integrity/ complete loss of protection.

Availability impact: None

We would argue that the impacted components, namely the windows devices, could have a total loss of availability. The loss of availability is dependent on how the attacker would exploit this vulnerability. Since the malicious server the attacker builds would be trusted by the impacted components, the attacker can inject anything to these devices, which means that they would have the ability to deny user's access to this resource.

References

https://www.youtube.com/watch?v=dCvB-mhkT0w

https://krebsonsecurity.com/2020/01/cryptic-rumblings-ahead-of-first-2020-patch-tuesday/#:~:text=Sources%20tell%20KrebsOnSecurity%20that%20Microsoft,in%20all%20versions%20of%20Windows.&text=This%20component%20was%20introduced%20into,back%20in%20Windows%20NT%204.0.

https://www.youtube.com/watch?v=8RI60aRyhoE

https://blog.qualys.com/vulnerabilities-research/2020/01/14/microsoft-windows-cryptoapi-spoofing-vulnerability-cve-2020-0601-how-to-detect-and-remediate

https://www.trendmicro.com/en_us/research/20/b/an-in-depth-technical-analysis-of-curveball-cve-2020-0601.html

http://cwe.mitre.org/data/definitions/295.html

https://github.com/ollypwn/CurveBall