SANS The most trusted source for cybersecurity training, certifications, degrees, and research

What you need to know about the Crypt32.dll / CryptoAPI Flaw

URGENT WEBCAST

Jake Williams
Senior Instructor
SANS Institute
@malwarejake

Johannes Ullrich
Instructor and Fellow
SANS Institute
@johullrich

Agenda

The vulnerability

What is ECC?

We use RSA, so we're good. Right?

Attack Scenarios

Why are we here?

On January 14, 2020 Microsoft patched a vulnerability involving certificate validation

The vulnerability (CVE-2020-0601) allows an attacker to bypass certificate validation, potentially leading to:

- **HTTPS man in the middle**
- **Bypass of some whitelisting controls**
- **Installation of malicious software updates**

According to Microsoft, the vulnerability only impacts ECC certificates

- **Certificates using RSA are not impacted**

Is all certificate validation on Windows broken?

Only certificate validation using the CryptoAPI is impacted

The following should not be vulnerable:

- **Some third-party software may use libraries that don't directly rely on the API and won't be vulnerable**
- **Certificate validation performed in the Windows kernel won't use CryptoAPI, which operates in user space**
- **Applications that perform certificate pinning (properly) should also not be vulnerable**
- **Even though the forged certificate will appear valid to the CryptoAPI, the fingerprint will not match the pinned certificate**
- **Windows updates are not vulnerable because they employ certificate pinning – this is probably a result of Flame**

What is ECC?

ECC – Elliptic Curve Cryptography

- **ECDSA (Elliptic Curve Digital Signature Algorithm) is favored over RSA for new applications owed to smaller key sizes**

While cracking an RSA key relies on factoring large numbers, cracking an ECDSA key requires solving the Elliptic Curve Discrete Logarithm Problem (ECDLP)

- **While mathematicians are making advances in factoring, no substantial advances have been made solving ECDLP**

ECDSA can use much smaller keys than RSA

- **256 bit ECDSA key provides about the same protection as a 3072 bit RSA key**

What if our software/app/whatever uses RSA certificates?

Early into the vulnerability release, some noted that only applications that use ECC certificates were vulnerable and any apps using RSA would be fine

While it is technically true that the parsing error only impacts ECC certificates, the certificates are being used in this case as a method of proving identity, making attacks more difficult

Attackers will replace the legitimate certificate with an ECDSA certificate, allowing them to successfully spoof (bypass) the identity check that is performed by the CryptoAPI

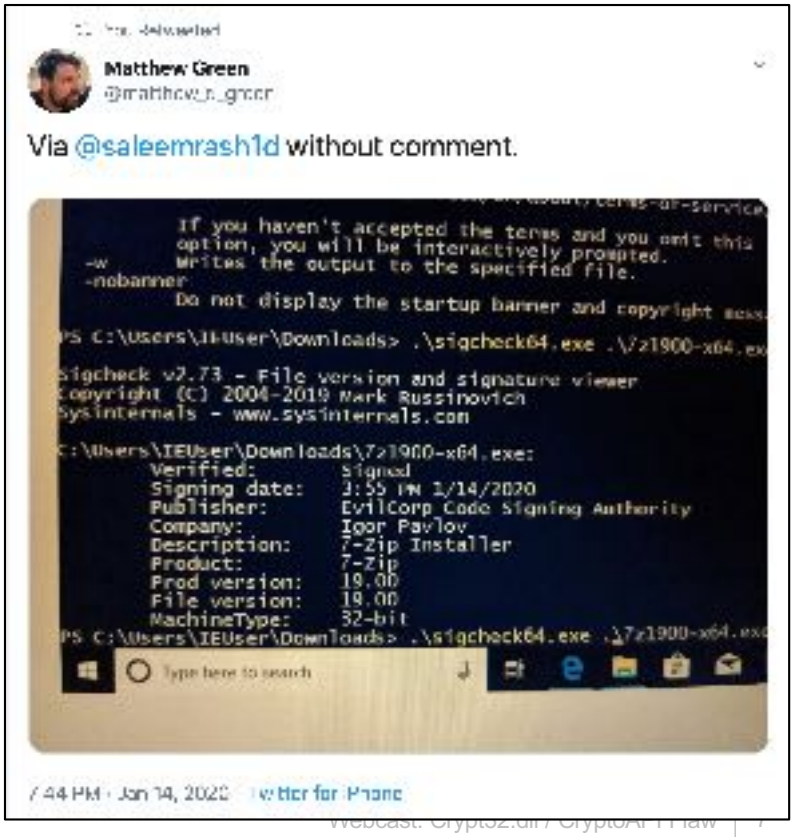
How quickly will CVE-2020-0601 be weaponized?

In their announcement, NSA stated that adversaries would be able to quickly weaponize the vulnerability

It looks like they weren't wrong…

It appears that an independent researcher was able to generate a working proof of concept in just a few hours

- Note – this could be a hoax where the user placed a rogue CA certificate in their root certificate store



Good News – Event Logs!

Microsoft created a function in Windows 10 called CveEventWrite

- The function is exported from Advapi32.dll

CveEventWrite facilitates the writing of a new log entry to the Application Event Log (though other log providers can be targeted)

Prior to the patch for 2020-0601, CveEventWrite was not imported into crypt32.dll, however the patched version uses this function

- The use of this function makes it somewhat easier for attackers to locate the vulnerable code since they can back up from the API call
- Attackers would discover the vulnerable code anyway – on balance, this is a significant benefit to defenders

Good News – Event Logs! (2)

Per the Microsoft documentation, the event logs will have the following attributes to query against:

- **Provider GUID: 85a62a0d-7e17-485f-9d4f-749a287193a6**
- **Source Name: Microsoft-Windows-Audit-CVE or Audit-CVE**
- **Message: will always contain "[CVE-2020-0601] cert validation"**

```
lea     r8, aCaSSha1SParaSO ; "CA: <%s> sha1: %s para: %s otherPara: %"...
mov     [rsp+170h+var_150], rax
mov     rdx, r15            ; unsigned __int64
mov     rcx, rbx            ; Dest
call    ?StringCchPrintfW@@YAJPEAG_KPEBGZZ ; StringCchPrintfW(ushort *,unsi
test    eax, eax
cmovns  r13, rbx

                            ; CODE XREF: ChainLogMSRC54294Error+1D7↑j
mov     rdx, r13
lea     rcx, aCve20200601Cer ; "[CVE-2020-0601] cert validation"
call    cs:__imp_CveEventWrite
```

Good News – Event Logs! (3)

Important: Note that no Application log entries will be written until the patch is applied

This is only useful in detecting attempted exploitation of a patched machine

- **e.g. A machine that isn't vulnerable**

Due to noise in Application event logs, many orgs don't forward them to the SIEM

- **After installing this patch, it's probably worth setting up a filter to forward "CVE" from Application logs to the SIEM**

Investigating Certificates

Per the NSA advisory, ECDSA certificates that use named elliptic curves "can be rule benign"

```
C:\certificates>certutil -asn ms-ecc-root-CA1.cer
0000: 30 82 03 23                                       ; SEQUENCE (323 Bytes)
0004:    30 82 02 a8                                    ; SEQUENCE (2a8 Bytes)
0008:    |  a0 03                                       ; OPTIONAL[0] (3 Bytes)
000a:    |  |  02 01                                    ; INTEGER (1 Bytes)
000c:    |  |     02
....
0188:    |  |  |     2b 81 04 00 22
         |  |  |        ; 1.3.132.0.34 ECDSA_P384 (secP384r1)
018d:    |  |  03 62                                    ; BIT_STRING (62 Bytes)
018f:    |  |     00
0190:    |  |     04 c7 11 16 2a 76 1d 56  8e be b9 62 65 d4 c3 ce
01a0:    |  |     b4 f0 c3 30 ec 8f 6d d7  6e 39 bc c8 49 ab ab b8
01b0:    |  |     e3 43 78 d5 81 06 5d ef  c7 7d 9f ce d6 b3 90 75
```

Investigating Certificates (2)

The NSA advisory notes that certutil can be used to list known curves

```
C:\certificates>certutil -displayEccCurve
Microsoft SSL Protocol Provider:
--------------------------------
Curve Name          Curve OID                    Public Key Length   CurveType          EccCurveFlags
-----------------------------------------------------------------------------------------------------
curve25519                                       255                 29                 0xa
nistP256            1.2.840.10045.3.1.7          256                 23                 0x7
nistP384            1.3.132.0.34                 384                 24                 0x7
brainpoolP256r1     1.3.36.3.3.2.8.1.1.7         256                 26                 0x7
brainpoolP384r1     1.3.36.3.3.2.8.1.1.11        384                 27                 0x7
brainpoolP512r1     1.3.36.3.3.2.8.1.1.13        512                 28                 0x7
```

Attack Scenarios

Most attack scenarios will involve some level of man in the middle (MITM) attack

Though MITM can happen anywhere, communications transiting areas of the world where ISP employees are a heightened insider threat risk are at elevated risk

We take our ISP communications relatively for granted in North America and Europe, but in many parts of the world, ISPs are considered part of the corporate threat model

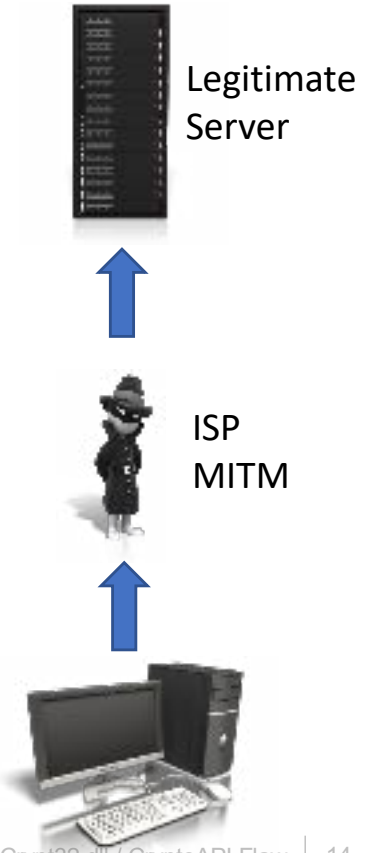
Road warriors using public wifi (including in-flight WiFi) are also at elevated risk

Attack Scenario #1 – Network Man in the Middle

The attacker is in a privileged position in the network and can intercept traffic between the vulnerable machine and the legitimate server

The attacker forges an ECDSA certificate for the legitimate site but the CryptoAPI validates it as a legitimate certificate

The attacker now has a full access to the plaintext traffic because they issue the symmetric keys used to protect session data

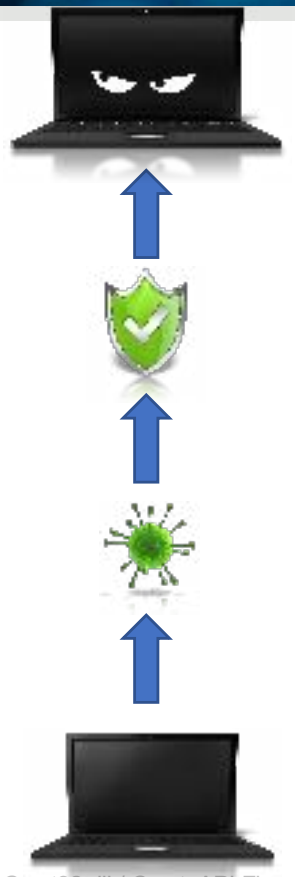


Attack Scenario #2 – Application Whitelisting Bypass

In this case, the attacker either has access to the vulnerable system or has delivered an executable to the target system

The attacker should not be able to execute a particular program because of application whitelisting that relies on certificates

Because the application whitelisting solution relies on the Windows CryptoAPI, the malicious application is allowed to execute without restrictions



Attack Scenario #3 – Malicious Software Update

The attacker is in a privileged position in the network and can intercept the request for a software update

- **The software update check is usually not user-initiated**

The attacker sends an "update" that is signed with a malicious certificate, passing validation

Even if the update is downloaded over HTTPS, if the CryptoAPI is used for communication (most likely), the attacker can MITM this as well (that requires two exploits, one for comms one for the installer)

Note: certificate pinning for the update check mitigates this

Legitimate Update Server

ISP MITM

Can I get a proof of concept to test my systems?

At this stage, multiple researchers have likely generated proof of concept exploits for this vulnerability

No responsible security professional will be sharing a weaponized PoC hours or days after a vulnerability like this is patched

I'm all for the release of offensive tools - a rising tide raises all ships

- **But it is FAR too early to release a tool for testing**

Closing Thoughts

This was bad, but far from as bad as it could have been

- **Nightmare scenarios would involve unsupported OS versions**

Kudos to NSA for turning this over to Microsoft to patch

Patch now and watch out for Application event log entries

The most obvious attack opportunities involve MITM, but other scenarios certainly are possible

- **This vulnerability is more likely to be used by APT attackers than less skilled attackers (aka script kiddies)**


SANS
URGENT
WEBCAST


SANS
URGENT WEBCAST