CVE-2024-31497

Magnus Opus Blueteam Security

Abstract

Kwetsbaarheden in cryptografie zijn een risico voor de integriteit van veilige communicatieprotocollen. Dit artikel richt zich op een recent ontdekte kwetsbaarheid in PuTTY, een van de meest gebruikte SSH en telnet clients. De exploit maakt gebruik van een voorspelbare cryptografische random nummer generator, wat de deur openzet voor potentiële aanvallen

Wynants Luka

s150425@ap.be

Inhoud

Inhoud

Versie beheer

versie	Datum	Aantal uur	Status	Wijziging
0.1.0	01/05/24	1	Initieel	Document lay-out in orde stellen
0.1.5	05/05/24	1	Inleiding	Inleiding aanmaken
0.2.0	05/05/24	3	Introductie	Introductie schrijven + research
0.3.0	10/05/24	2	Research	Research over ECDSA, kwetsbaarheiden in ECDSA, lattice attacks op zwakke random number generators
0.3.5	11/05/24	1	Voorgrondkennis	Voorgrondkennis schrijven
0.4.0	13/05/24	3	Termen	Voorgrondkennis+Termen schrijven
0.5.0	16/05/24	3	Identify	Identify + het probleem beschrijving schrijven
0.5.5	17/05/24	2	identify	Identify afmaken
0.6.0	22/05/24	2	Protect + Research	Protect gedeelte schrijven + research
0.7.0	29/05/24	4	Detect+ Research	Detect schrijven + research en Wazuh research
0.8.0	30/05/24	2	Respond	Respond gedeelte schrijven
0.9.9	31/05/24	1	Respond	Respond gedeelte schrijven
0.9.5	01/06/24	1	POC	Proof of concept uitvoeren
1.0.0	01/06/24	1	finaal	Finale aanpassingen, spelchecken

1.0 INTRODUCTIE

PuTTY is een gratis en open-source SSH (Secure Shell) en Telnet-client, ontwikkeld door Simon Tatham. Het fungeert als terminal emulator, seriële console en netwerkfile transfer applicatie. Daarnaast biedt PuTTY de mogelijkheid om SSH-sleutelparen te generen en digitale handtekeningen te ondertekenen. Hier blijkt de kwetsbaarheid te liggen: de ECDSA-implementatie, een digitale handtekening algoritme blijkt kwetsbaar te zijn door een bias nonce generatie, die ertoe leidt dat je privésleutel wordt berekend na analyse van ongeveer 58 handtekeningen.

Link naar CVE-rapport: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2024-31497

1.1 Samenvatting van de Inhoud

In dit artikel zullen we een diepgaande dreigingsanalyse van de exploit uitvoeren, de risico's identificeren die aan de exploit zijn verbonden en een proof of concept uitvoeren, waarbij de mogelijke aanvalsvectoren worden beschreven. Hiernaast zullen we het ook hebben over hoe je jezelf kan beschermen tegen deze kwetsbaarheid en hoe je kan weten of je kwetsbaar bent.

1.2 Algemene informatie

De exploit is ontdekt begin april 2024 door Fabian Bäumer en Marcus Brinkmann van de Ruhr Universiteit Bochum. PuTTY versies 0.68 tot en met 0.80 bleken kwetsbaar te zijn, dit omvat 7 jaar van putty versies.

1.2.1 Wordt hij al in het wild gebruikt?

De exploit vindt zijn oorsprong in een slechte ECDSA-implementatie. Hoewel dit nog niet gerapporteerd is voor PuTTY handtekeningen is deze kwetsbaarheid al lang bekend, vooral in zwak gegenereerde Bitcoin handtekeningen die ook ECDSA implementeren. Dit is uitgebreid beschreven in de volgende artikelen en video:

- https://eprint.iacr.org/2019/023.pdf
- https://eprint.iacr.org/2020/1540.pdf
- https://cryptopals.com/sets/8/challenges/62.txt
- https://www.youtube.com/watch?v=6ssTlSSIJQE

2.0 Voorgrondkennis

2.1 Elliptische curves en groep theorie

Een elliptische curve is een curve die bestaat uit punten die voldoen aan de volgende vergelijking:

$$y^2 = x^3 + ax + b$$

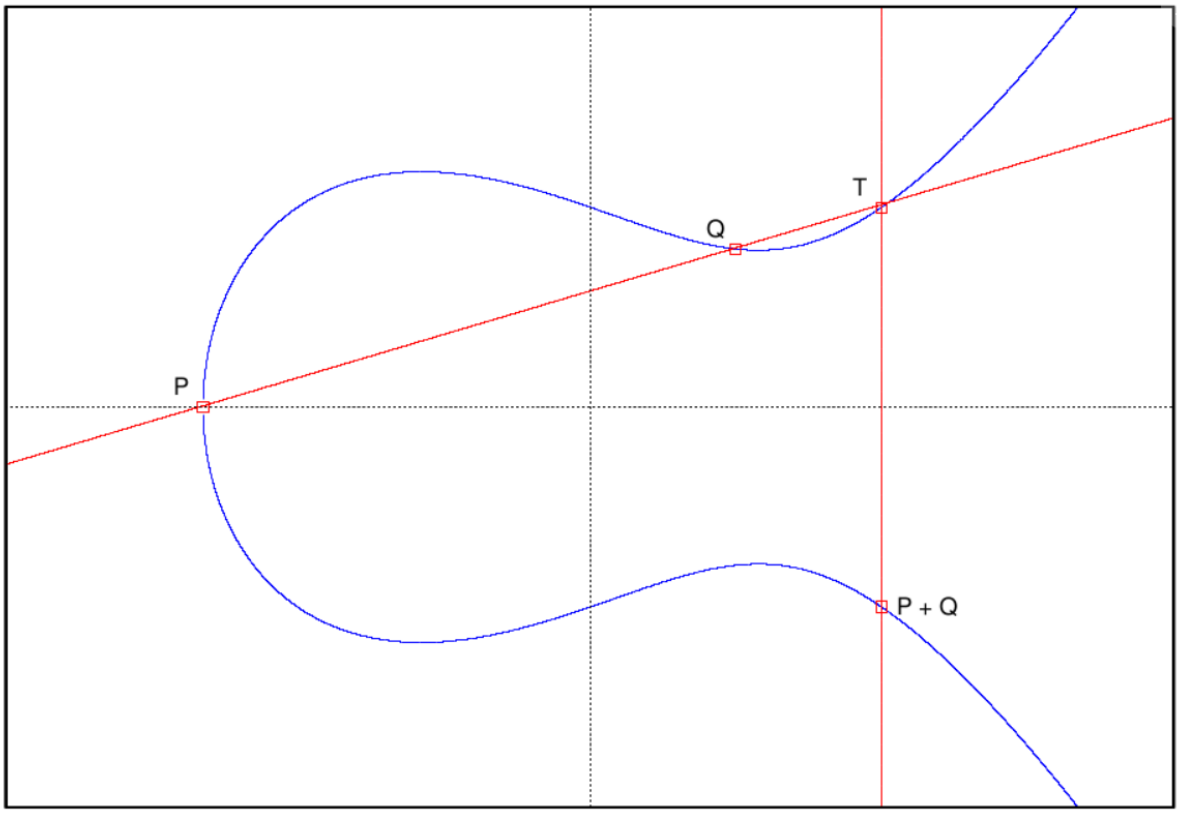


Fig. 1.0 Elliptische curve [30]

De groepswet geldend voor een elliptische curve biedt een manier om punten op de curve te genereren uit andere punten. Als voorbeeld, als je het punt P verdubbeld is het nog steeds een punt op de curve, dit wordt scalaire multiplicatie genoemd. [11]

Het generatorpunt G is het startpunt voor alle andere punten op de elliptische curve, in cryptografie wordt het gebruikt om de publieke sleutel te berekenen.

Het identity element is een punt op de curve dat het einde van optelling markeert.

Men kan het generator punt G met zichzelf optellen tot het identity element is bereikt. Het aantal verschillende punten dat hierbij gegenereerd wordt wordt het order van de elliptische curve genoemd. In cryptografische standaarden is het belangrijk om een heel groot priemgetal te gebruiken als generator punt.

2.2 Termen

Crypto RNG: Cryptografische random nummer generator

Elliptische curve-cryptografie (ECC): Elliptic Curve Cryptografie (ECC) is een public-key cryptografie methode dat gebruik maakt van de algebraïsche structuur van elliptische curven. Het biedt hoge beveiliging met relatief kleine sleutels in vergelijking met andere methoden zoals RSA.

Eliptic Curve Digital Signature Algorithm (ECDSA): Een algoritme dat gebruik maakt van een elliptic curve om data te ondertekenen met een digitale handtekening

NIST P521 ECDSA: Een ECDSA algoritme dat de secp521r1 curve gebruikt voor handtekening te ondertekenen. [16]

Nonce: Een nonce (afkorting van "number used once") is een willekeurig gegenereerd getal dat bij elke digitale handtekening in het ECDSA-proces wordt gebruikt. Het is belangrijk dat een nonce:

- uniform willekeurig wordt gegenereerd
- zijn bitlengte overeenkomen met de bitlengte van de order van de curve [20]

In deze paper zal ik de soms de letter k gebruiken om te verwijzen naar een nonce.

2.3 ECDSA signature proces

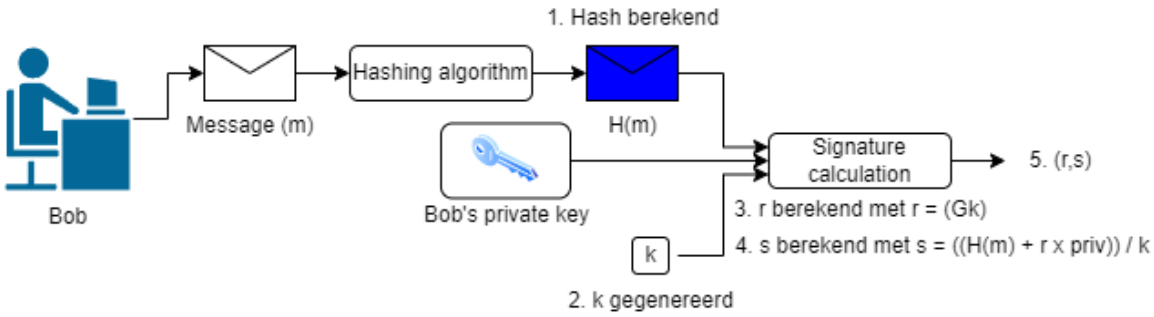


Fig. 1.1 ECDSA signature process [9]

Om een handtekening te maken van een bericht m met ECDSA wordt de volgende procedure gevolgd:

1. Eerst wordt er een hash berekend van het originele bericht: $h = H(m)$
2. Een willekeurig getal k wordt gegenereerd
3. De r-waarde wordt berekend, wat de x-coördinaat is van: $r = (Gk)_x$
4. Hierna wordt de s-waarde berekend met $s = k^{-1}(H(m) + r \times priv)$
5. De handtekening is dan de tuple (r, s)

[9] [18]

2.3.1 Zwakte punten in ECDSA

De veiligheid van het ECDSA hangt af van de geheimhouding en onvoorspelbaarheid van de gegenereerde nonce.

Als de nonce k gekend is, kan de privé sleutel eenvoudig worden berekend met simpele algebraïsche herschikking van het s-waarde formule:

$$s = k^{-1}(H(m) + r \times priv)$$

$$sk = H(m) + r \times priv$$

$$sk - H(m) = r \times priv$$

$$priv = r^{-1}(sk - H(m))$$

Hergebruik van dezelfde nonce k in 2 verschillende handtekeningen maakt het ook mogelijk om de privé sleutel te berekenen. Je kan dit doen door de volgende 2 s-waarde vergelijkingen te maken van 2 digitale handtekeningen:

$$s_1 = k^{-1}(h_1 + r \times priv) \text{ en } s_2 = k^{-1}(h_2 + r \times priv)$$

Omdat de waarde van de nonce k, de privésleutel en r hetzelfde zijn in beide s-waardes kan je de nonce berekenen door ze van elkaar af te trekken:

$$s_1 - s_2 = k^{-1}(h_1 + r \times priv) - k^{-1}(h_2 + r \times priv)$$

$$s_1 - s_2 = k^{-1}(h_1 - h_2)$$

$$k(s_1 - s_2) = h_1 - h_2$$

$$k = (h_1 - h_2)(s_1 - s_2)^{-1}$$

Nu kan je de formule van hierboven toepassen om de privé sleutel te calculeren. Kan je al zien waarom het zo belangrijk is om een crypto RNG te gebruiken tijdens het genereren van een nonce?

Hoewel als een slechte crypto RNG gebruikt is kan een nonce ook voorspelbaar zijn. Een voorbeeld hiervan is door een aantal digitale handtekeningen te analyseren zodat met behulp van Lattice attacks de privé sleutel kan berekend worden. Lattice attacks zijn gevaarlijk voor ECDSA omdat een paar bit nonce-lekken in meerdere handtekeningen voldoende zijn om de geheime sleutel te berekenen. Momenteel wordt het BKZ-algoritme en LLL-algoritme vaak gebruikt als aanval technieken.[21]

Zie: https://cryptopals.com/sets/8/challenges/62.txt [8], voor de wiskunde hierachter te bekijken.

3.0 IDENTIFY

3.1 Wat is exact het veiligheidsprobleem?

De gekozen kwetsbaarheid voor deze exploit evaluatie is een zwakte in PuTTY. Deze versies van PuTTY bevatten een biased crypto RNG voor ECDSA nonce generatie. De willekeurigheid van een nonce is cruciaal voor het garanderen van de secrecy van de privé sleutel.

3.1.1 Hoe wordt de nonce bij PuTTY gegenereerd?

We hebben eerder gezien dat de bitlengte van een nonce moet overeenkomen met de bitlengte van het de order van de curve anders voldoet dit niet aan de standaarden.

PuTTY gebruikt $SHA\text{-}512(ID||SHA\text{-}512(x)||SHA1(m))modulo$ als een pseudo-willekeurige nonce.[4] Dit betekent dat nonces genereerde zijn in het bereik:

$[1, 2^{512}]$

in plaats van het bereik:

$[1, n-1]$, Waarbij $n \approx 2^{521}$ dus $[1, 2^{521}-1]$,

Dit komt doordat een SHA-512 hashing algoritme wordt gebruikt en zal een 512-bit nummer teruggeven. [22] Hierdoor is elke gegenereerde nonce maar 512-bit random en worden er 9 nullen toegevoegd aan de nonce. Met deze bias en lattice attacks kan een privésleutel worden berekend van ongeveer $521 \div 9 \approx 58$ digitale handtekeningen.[15]

Fig. 1.2 PuTTY voorspelbare nonce illustratie

3.2 Hoe kan de exploit uitgevoerd worden?

3.2.1 PuTTY Pageant

Pageant dient als een sleutelbeheerder die uw SSH-sleutels bewaart en ze aan PuTTY doorgeeft wanneer dat nodig is voor verificatie. Hier is de basic flow van hoe Pageant wordt gebruikt om een verbinding te maken met een SSH-server:

1. De client vraagt om een verbinding te maken met de SSH-server door een publieke sleutel voor te leggen.
2. De server genereert en verstuurt een kort, willekeurig bericht en vraagt de client om het te ondertekenen met de privésleutel.
3. De cliënt stuurt het bericht door naar de SSH Pageant en vraagt om het te ondertekenen
4. De Pageant ondertekend het bericht en stuurt het resultaat terug naar de client die het doorstuurt naar de server.
5. De server controleert de handtekening met de publieke sleutel van de client.
6. De server heeft nu bewijs dat de cliënt in het bezit is van zijn privésleutel.

[19]

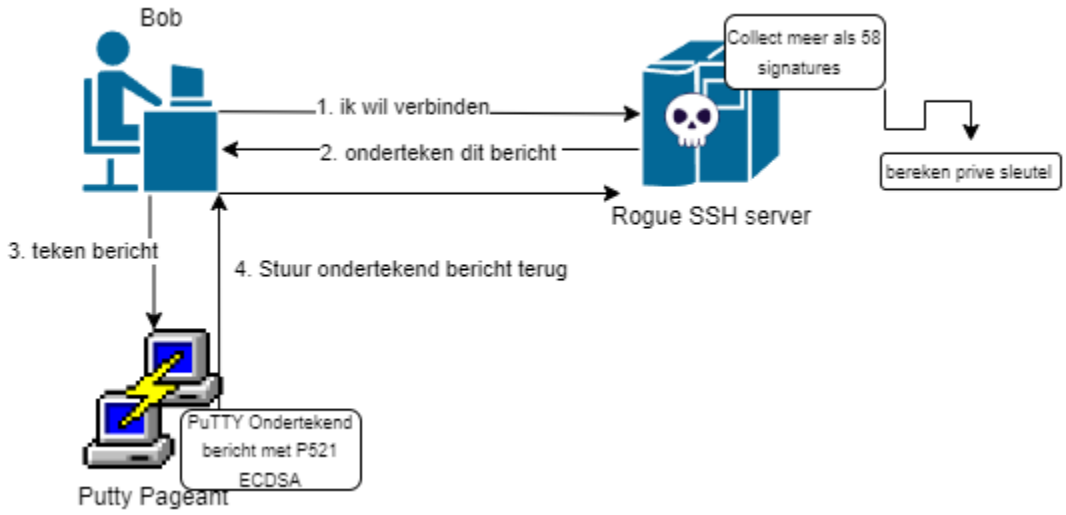


Fig. 1.3 ECDSA signature proces

Het risico ontstaat wanneer er meerdere malen verbinding wordt gemaakt met een compromised of rogue SSH-server. In zo'n situatie kan de server ongeveer 58 handtekeningen verzamelen en hiervan de privésleutel berekenen.

3.2.2 Git commits

Een 2de mogelijke attack vector is door ECDSA-handtekeningen van meer als 58 geverifieerde commits in GitHub of andere publieke repositories te collecteren. Als deze handtekeningen ondertekend zijn met PuTTY kan een aanvaller de privé sleutel berekenen.

3.3 Wat is daar gevaarlijk aan?

3.3.1 Privé sleutel wordt brekend

Het grootste risico van deze exploit is dat iemand buiten je organisatie jouw privésleutel kan achterhalen met behulp van ongeveer 58 digitale handtekeningen en wat wiskunde. Als ze jouw privésleutel hebben, kunnen ze verschillende ongeautoriseerde acties uitvoeren.

3.3.2 Supply Chain Attacks:

De exploit maakt supply chain attacks mogelijk doordat aanvallers ondertekende berichten kunnen stelen van applicaties, zoals FileZilla, WinSCP TortoiseGit, PuTTY pageante en XenCenter. Deze software gebruikt kwetsbare versies van PuTTY. Aanvallers kunnen de kwetsbaarheid misbruiken om NIST P521 prive sleutels te kraken die door deze softwarecomponenten worden gebruikt.

3.4 Welk risico brengt dit mee voor je organisatie?

3.4.1 Privé sleutels gebruiken voor ongeautoriseerde toegang en controle:

Als de privé sleutel wordt gebruikt bij SSH-authenticatie kunnen aanvallers zich voordoen als legitieme gebruikers en toegang krijgen tot kritieke systemen en applicaties binnen het netwerk van je organisatie. Dit kan leiden tot datalekken, inbreuk op systemen en ongeautoriseerde wijzigingen van gevoelige gegevens of configuraties. Aanvallers kunnen misbruik maken van overgenomen accounts om privileges te escaleren en over te schakelen naar andere systemen.

3.4.2 Impersonatie

Een aanvaller kan geldige digitale handtekeningen op berichten vervalsen, waardoor het lijkt alsof ze zijn verzonden door de legitieme persoon in je organisatie. Dit kan ertoe leiden dat misinformatie of ongeautoriseerde instructies worden geaccepteerd.

3.4.3 ongeautoriseerde commits toevoegen aan code

Als de privésleutel wordt gebruikt om software-updates of commits te ondertekenen, kan de aanvaller malware verspreiden die zich voordoet als een legitieme update.

Hij kan deze commits maken lokaal in je organisatie als hij backdoor access heeft, of op publieke repositories zoals GitHub.

Dit is een mogelijke supply chain attack op organisaties die software ontwikkelen. Hierdoor kan de organisatie ook reputatieschade krijgen door het verspreiden van geïnfecteerde software.

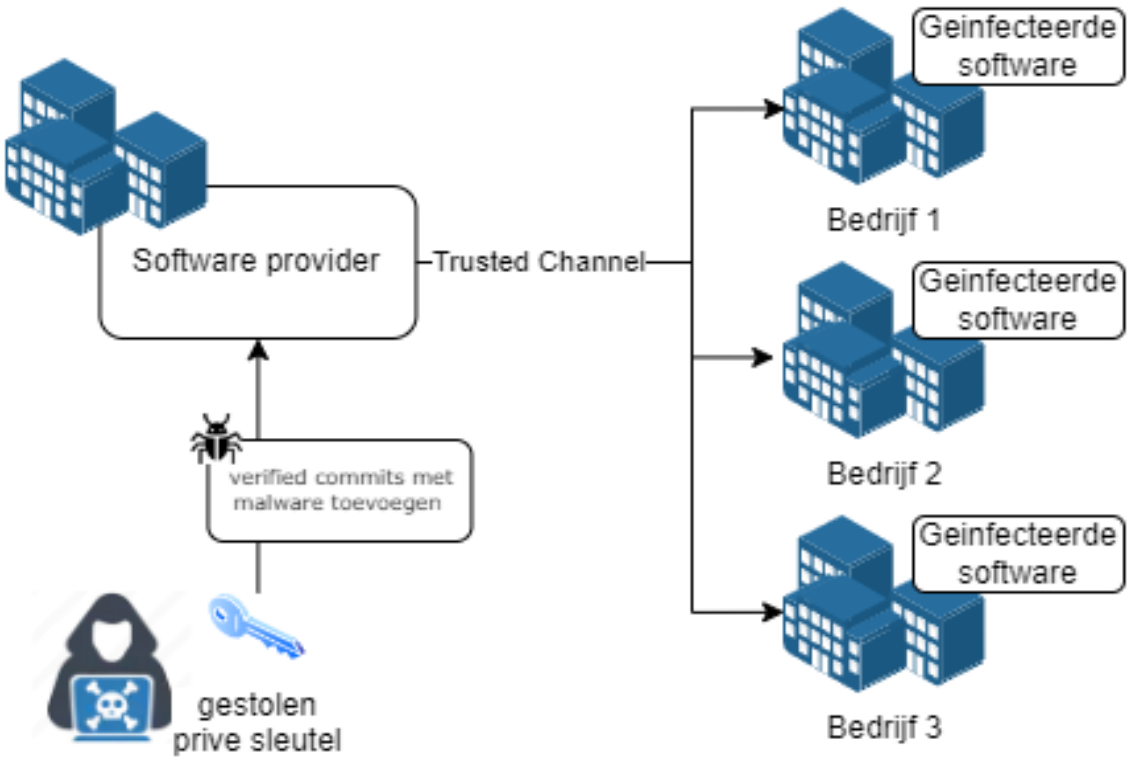


Fig. 1.4 Supply Chain Attack

3.5 Hoe kan deze binnenkomen?

3.5.1 Gekende software dat kwetsbare PuTTY software gebruikt:

- PuTTY versies voor 0.81
- FileZilla versies 3.24.1 tot 3.66.5
- WinSCP versies 5.9.5 tot 6.3.2
- TortoiseGit versies 2.4.0.2 tot 2.15.0
- TortoiseSVN versies 1.10.0 tot 1.14.6
- XenCenter versies 8.2.6
- PuTTY Pageant als de SSH server onbetrouwbaar is

4.0 POC

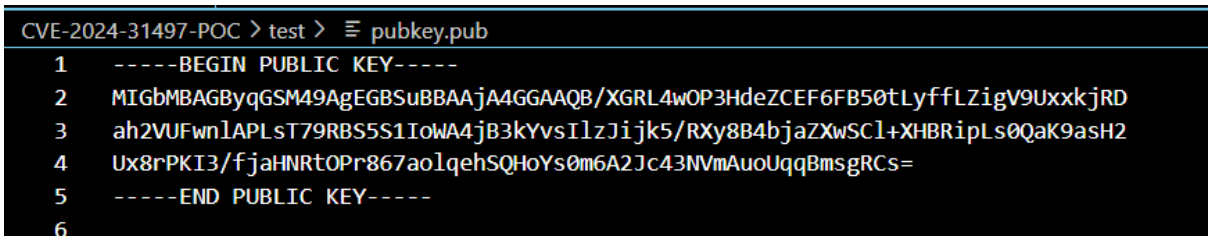
Voor deze Proof of Concept uit te voeren kan je de volgende github repository gebruiken:

https://github.com/HugoBond/CVE-2024-31497-POC/tree/main

Om de POC uit te voeren, genereer je eerst een aantal digitale handtekeningen met een voorspelbare nonce en een public key, door test.py uit te voeren.

Dit zijn bijvoorbeeld 60 digitale handtekeningen gegenereerd met een voorspelbare nonce:

Volgend public key van de repository zou hier voor gebruikt kunnen worden:



Ik kon spijtig genoeg niet de rest van de POC uitvoeren. Het installeren van de nodige libraries, sage-math en fplll genereerde een error.

5.0 PROTECT

5.1 Hoe ga je je infrastructuur beschermen tegen deze exploit?

5.1.1 Betreffende software updaten:

Zorg ervoor dat alle kwetsbare software, zoals PuTTY, FileZilla, WinSCP TortoiseGit, XenCenter zijn geüpdatet naar versies die de kwetsbaarheid hebben gepacht. Update PuTTY naar versie 0.81, FileZilla naar versie 3.67.0, WinSCP naar versie 6.3.3, TortoiseGit naar versie 2.15.0.1 en XenCenter naar versie 8.2.7.

5.1.2 Regenereer sleutels

Controleer alle NIST P-521-sleutels die worden gebruikt met PuTTY of in andere kwetsbare software. Deze sleutels moeten onmiddellijk worden ingetrokken en opnieuw worden gegenereerd. Dit zorgt ervoor dat, zelfs als de exploit is gebruikt om de sleutels te berekenen, de betreffende sleutels niet meer geldig zijn.

5.1.3 Gebruik verschillende sleutels

Gebruik aparte sleutelparen voor verschillende operaties. Dit wordt gezien als best practice in security omdat het de veiligheid verbetert door ervoor te zorgen dat het kraken van één sleutel geen invloed heeft op alle operaties.

5.1.4 Monitoring en detectie implementeren:

Monitor en detect verdachte activiteiten door het controleren op abnormale login pogingen via SSH en ongewone gebruik van prive sleutels op rare tijdstippen. Gebruik een Security information and event manager (SIEM) Wazuh met agents om dit te automatiseren.

5.1.5 Gebruikers opleiden:

Informeer gebruikers over het belang van key security en de risico's die daaraan verbonden. Moedig gebruikers aan om best practices voor sleutelbeheer te volgen, zoals het regelmatig regenereren van sleutels.

5.2 Hoe kan de exploit tegenhouden?

5.2.1 Veilige Nonce-generatie implementeren:

Zorg ervoor dat software veilige methodes gebruikt voor nonce’s te generen. Om deze exploit te vermijden, is het van cruciaal belang om de standaarden te volgen dat in dit document worden gespecifieerd voor P-521 ECC:

https://www.secg.org/sec2-v2.pdf#subsubsection.2.6.1

6.0 DETECT

6.1 Hoe kan ik zien of een systeem kwetsbaar is voor de exploit?

Stap 1: Controleer of op uw systeem een van de kwetsbare softwareversies gebruikt. Voorbeelden van getroffen software om te checken zijn:

- PuTTY-versies van 0.68 tot 0.80
- FileZilla-versies van 3.24.1 tot 3.66.5
- WinSCP-versies van 5.9.5 tot 6.3.2
- TortoiseGit versies van 2.4.0.2 naar 2.15.0
- TortoiseSVN versies van 1.10.0 tot 1.14.6

Stap 2: Kijk na of je ooit NIST P-521 sleutels hebt gebruikt in PuTTY of getroffen software

De sleutelaanmaak in PuTTY voor NIST P-521 sleutels is niet kwetsbaar, de kwetsbaarheid ligt bij het ondertekenen van handtekeningen met PuTTY. Dus als je ooit een NIST P-521 sleutel hebt gebruikt om met PuTTY iets te ondertekenen, is het het beste om deze sleutel te vernieuwen.

Stap 3: Beoordeel hoe deze sleutels worden gebruikt in uw omgeving, als voorbeeld:

- Controleer of de sleutels gebruikt worden voor SSH-authenticatie naar externe servers.
- Controleer of deze sleutels gebruikt worden voor het ondertekenen van data/commits

6.2 Hoe kan ik detecteren dat de exploit gebruikt wordt:

6.2.1 SSH-verkeer controleren op abnormale logins

Kijk naar abnormale login patronen of succesvolle aanmeldingen via SSH vanaf ongewone IP-adressen of op vreemde uren. Geautomatiseerde tools zoals Wazuh kunnen helpen deze patronen te identificeren.

6.2.2 Git commits controleren:

Als je organisatie Git gebruikt voor versiebeheer, controleer dan de repositories op het gebruik van NIST P-521 sleutels in commit message handtekeningen. Kijk of de commits effectief getekend zijn door mensen uit je organisatie. Tools zoals GitHub bieden logs en meldingen voor commit activiteiten. Controleer regelmatig openbare repositories van je organisatie.

6.2.2 Yara Rule integreren in Wazuh

YARA (Yet Another Recursive Acronym) is een hulpmiddel voor het identificeren en classificeren van malware of andere specifieke bestanden op basis van patronen of kenmerken in hun inhoud. [25] Je kunt YARA integreren in je Wazuh omgeving om te scannen voor kwetsbare software, volg de guide van de Wazuh documentatie om YARA rules te integreren met gebruik van de volgende YARA rule:

```
rule CVE_2024_31497_Putty_ECSDA
{
    meta:
        author="Matt Richard (mjr@stairwell.com)"
        date="2024-04-17"
        description="Finds putty weak ecc-ssh ECDSA generator CVE-2024-31497"

reference="https://git.tartarus.org/?p=simon/putty.git;a=commit;h=c193fe9848f50a88a4089aac
647fecc31ae96d27"
        hash="b0df9f1c67b830d5c4e2c88565e6c12b08a6634a3c1d234627ffa95fb5e8bcef"
        hash="a78f4a4937ee5aca61d66bc5168b6becbd7c273a37e0ec5f8a42818ac61e1e34"
        hash="512e27ef54ccaca2dded62e43b7983bff7c29ef911ce504d099253ff03ef73da"
        hash="71cf414127887dcc4e5282bafd6e67a7dd1840d6d25a03af1664eceda2c6816a"
        hash="33971e14b7dd85326bc6d50aca9dd506e68e9104f2811dcbd07fe92bfcbcbb50"
        hash="b830f9c9130c60f31829707ce94dd868db458494af3d9d9b380909ebd32c7ddd"
        hash="eb1b278b91a8f183f9749948abd9556ec21b03ca852c53e423d824d5d7cc3de4"
        hash="97917d2459a395fde40c70bebe1b133624d83ad875d44164b940015996ef7ceb"
    strings:
        $dsa_str = "ECDSA deterministic k generator"
    condition:
        $dsa_str
}
```

[6]

6.2.3 Unusual SSH logins monitoren met Wazuh

Je kan ook verdachte SSH-logins controleren met Wazuh, voeg de SSH-logfile toe aan de Wazuh agent configuratie file, ossec.conf:

```
<localfile>
  <log_format>syslog</log_format>
  <location>/var/secure/auth.log</location>
</localfile>
```

[26]

Definieer nu rules in het local_rules.xml file om abnormale SSH-activiteiten te detecteren. Abnormale activiteiten die je kan detecteren zijn mislukte login pogingen, login pogingen vanaf onbekende IP-adressen of login pogingen tijdens ongewone uren. Een voorbeeld van zo een rule kan zijn:

```
<rule id="100003" level="10">
  <if_group>authentication_success</if_group>
  <time>8:00 pm - 8:00 am</time>
  <description>Successful login during non-business hours. Id=</description>
  <group>login_time</group>
</rule>
```

Deze rule creëert een level 10 event op logins tussen 20:00 uur - 8:00 uur.

7.0 RESPOND en RECOVER

7.1 Wat ga je doen als een exploit gebruikt is?

1. Sleutels uitschakelen:

Trek onmiddellijk alle gecompromitteerde NIST P-521 sleutels in en verwijder alle systemen en diensten (bijv. verwijderen uit authorized_keys bestanden, toegang intrekken in GitHub, etc.).

2. Vervang alle NIST P-521 sleutels:

Genereer nieuwe ECDSA-sleutels met een veilig algoritme. Zorg ervoor dat oude sleutels volledig verwijderd en ongeldig gemaakt zijn.

3. Veilige configuratie:

Controleer en verbeter de configuratie van SSH-diensten en zorg ervoor dat ze de beste beveiligingspraktijken worden gevolgd.

4. Software nakijken:

Dubbel check dat alle software die de vulnerability bevat is geupdate, gebruik scanners of YARA rules voor de hashes van alle software te vergelijken met de hashes van de kwetsbare software.

7.1.1 Als de exploit ernaar leidt dat systemen gecompromitteerd zijn:

1. Isoleer gecompromitteerde systemen:

Koppel de besmette systemen los van het netwerk om verdere exploitatie te hinderen.

2. Logs analyseren:

Verzamel en analyseer logs van besmette systemen om de schaal en tijdlijn van de inbreuk te bepalen. Zoek naar bewijs van ongeautoriseerde toegang of verdachte activiteiten.

3. Controleer de gecompromitteerde systemen:

Check of er geen backdoors of andere malware/virussen zijn achtergelaten door de aanvaller. Dit betekent ook het controleren van opstartscripts, cronjobs en services. Voer regelmatig virus of malwaredetectie uit op de systemen

4. Bepaal de impact:

Bepaal of er toegang is verkregen tot gevoelige gegevens of dat deze zijn geëxfiltreerd. Hiervoor kan het nodig zijn om netwerkverkeer en logs van systemen te bekijken.

7.2 Wat kan de impact zijn?

De mogelijke impact van het exploit:

- Private key exposure
- Toegang tot gevoelige gegevens
- Supply Chain Attacks
- Ongeautoriseerde access aan systemen/infrastructuur
- Financiële verlies

9.0 Bronnen

[1]	https://blog.trailofbits.com/2020/06/11/ecdsa-handle-with-care/
[2]	https://www.chiark.greenend.org.uk/~sgtatham/putty/wishlist/vuln-p521-bias.html
[3]	https://www.cert.be/nl/advisory/warning-vulnerability-affecting-putty-client
[4]	https://twitter.com/lambdafu/status/1779969509522133272
[5]	https://eprint.iacr.org/2020/1540
[6]	https://stairwell.com/resources/stairwell-threat-report-vulnerable-putty-ssh-libraries-cve-2024-31497/
[7]	https://www.openwall.com/lists/oss-security/2024/04/15/6
[8]	https://cryptopals.com/sets/8/challenges/62.txt
[9]	https://asecuritysite.com/signatures/ecd5
[10]	https://asecuritysite.com/encryption/ecd4
[11]	https://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/
[12]	https://www.youtube.com/watch?v=6ssTlSSIJQE
[13]	https://tches.iacr.org/index.php/TCHES/article/view/8684/8243
[14]	https://www.putty.org/
[15]	https://crypto.stackexchange.com/questions/111474/cve-2024-31497-nonces-and-random-numbers-can-someone-explain-please
[16]	https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Standards-and-Guidelines/documents/examples/P521_SHA512.pdf
[17]	https://cryptobook.nakov.com/asymmetric-key-ciphers/elliptic-curve-cryptography-ecc
[18]	https://link.springer.com/chapter/10.1007/978-3-030-00470-5_29
[19]	https://smallstep.com/blog/ssh-agent-explained/
[20]	https://www.secg.org/sec2-v2.pdf#subsubsection.2.6.1
[21]	https://eprint.iacr.org/2021/1489.pdf
[22]	https://stackoverflow.com/questions/18236106/what-is-the-length-of-a-hashed-string-with-sha512
[23]	https://git.tartarus.org/?p=simon/putty.git;a=blobdiff;f=crypto/ecc-ssh.c;h=5fa25189a55dc7e2517d73e0cbf3651007939cca;hp=d31978669cc5b

	58405ac092e9cc0a9993d92d4b1;hb=c193fe9848f50a88a4089aac647fecc31ae96d27;hpb=aab089267118feb50c7bb1a2a68f96d9b097daa2
[24]	https://documentation.wazuh.com/current/proof-of-concept-guide/detect-malware-yara-integration.html
[25]	https://virustotal.github.io/yara/
[26]	https://documentation.wazuh.com/current/user-manual/reference/ossec-conf/localfile.html
[27]	https://github.com/HugoBond/CVE-2024-31497-POC
[28]	https://www.cyfrin.io/blog/elliptic-curve-digital-signature-algorithm-and-signatures#:~:text=1.-,ECDSA%20Key%20Generation%20%2D%20How%20are%20Public%20and%20Private%20Keys%20Created,elliptic%20curve%20pubKey%20%3D%20p%20*%20G
[29]	https://cryptobook.nakov.com/asymmetric-key-ciphers/elliptic-curve-cryptography-ecc
[30]	https://www.quora.com/What-is-the-equation-of-an-elliptic-curve-y%C2%B2-x%C2%B3-ax-b
[31]	chatgtp voor grammatische checking