Nevada Ransomware, Nokoyawa Variant | ThreatLabz

zscaler.com/blogs/security-research/nevada-ransomware-yet-another-nokoyawa-variant

Brett Stone-Gross

Concerned about VPN vulnerabilities? Learn how you can benefit from our VPN migration offer including 60 days free service.

Talk to an expert

Zscaler Blog

Get the latest Zscaler blog updates in your inbox

Subscribe

Security Research

Key Takeaways

- *Nevada* ransomware was advertised in criminal forums in December 2022 as part of a new ransomware-as-a-service affiliate program
- Nevada is written in the Rust programming language with support for Linux and 64-bit versions of Windows
- Zscaler ThreatLabz has identified significant code similarities between Nevada and *Nokoyawa* ransomware including debug strings, command-line arguments and encryption algorithms
- The Nokoyawa ransomware codebase has been continuously modified with at least four distinct variants (including Nevada) that have emerged since February 2022
- The Nokoyawa threat group appears to operate two parallel code branches written in different programming languages designed to confuse researchers and evade detection

Zscaler ThreatLabz has been tracking the Nokoyawa ransomware family and its predecessors including Karma and Nemty ransomware. The original version of Nokoyawa ransomware was introduced in February 2022 and written in the C programming language. File encryption utilized asymmetric Elliptic Curve Cryptography (ECC) with Curve SECT233R1 (*a.k.a.* NIST B-233) using the Tiny-ECDH open source library combined with a per file Salsa20 symmetric key. In September 2022, a Rust-based version of Nokoyawa ransomware was released. This new version used Salsa20 for symmetric encryption, but the ECC algorithm was replaced with Curve25519. In December 2022, Nevada ransomware was advertised in criminal forums. ThreatLabz has determined that Nevada shares significant code with the Rust-based variant of Nokoyawa. In January 2023, ThreatLabz also identified another version of Nokoyawa written in C that is similar to the original version, but uses the same configuration options (passed via the command-line) as the Rust-based Nokoyawa 2.0.

In this blog, we analyze Nevada ransomware and how it compares to the other versions of Nokoyawa ransomware. Based on the numerous similarities, the Nokoyawa threat group appears to utilize two separate branches for ransomware attacks.

Technical Analysis

ThreatLabz has identified at least four distinct versions of Nokoyawa ransomware. For clarity, we will use the version numbers 1.0, 1.1, 2.0 and 2.1 (Nevada) based on code similarities. Table 1 illustrates the similarities and differences between all four versions of Nokoyawa ransomware including Nevada.

Attribute	Nokoyawa 1.0	Nokoyawa 1.1	Nokoyawa 2.0	Nokoyawa 2.1 (Nevada)
Encryption algorithms	SECT233R1 + Salsa20	SECT233R1 + Salsa20	X25519 + Salsa20	X25519 + Salsa20
Encryption library	Tiny-ECDH	Tiny-ECDH	x25519_dalek	x25519_dalek
Programming language	C/C++	C/C++	Rust	Rust
Encryption Parameters	Hardcoded	Passed via command-line	Passed via command-line	Hardcoded
Import Hashing	No	Yes	No	No
CIS Exclusion	No	No	Yes	Yes
Architecture	x64	x64	x64	x64
Earliest known compilation date	February 2022	January 2023	September 2022	January 2023

Table 1. Comparison between different versions of Nokoyawa ransomware

There are a few commonalities between all Nokoyawa variants such as being compiled only for 64-bit versions of Windows and using a relatively obscure method to delete Windows Shadow Copies. The latter entails calling the function *DeviceIoControl* (shown in Figure 1) with the undocumented control code parameter *IOCTL_VOLSNAP_SET_MAX_DIFF_AREA_SIZE* (0x53C028) with a maximum size of 1, which causes Windows to delete all shadow copies as a result.

```
loc_140005F1E:                          ; CODE XREF: DeleteShadowCopies_0:DeleteShadowCopies↑j
            lea     rax, [rbp+70h+BytesReturned]
            mov     [rsp+0F0h+var_C0], rax ; lpBytesReturned
            mov     [rsp+0F0h+var_B8], 0 ; lpOverlapped
            mov     [rsp+0F0h+nOutBufferSize], 0 ; nOutBufferSize
            mov     qword ptr [rsp+0F0h+var_D0], 0 ; lpOutBuffer
            lea     r8, [rbp+70h+InBuffer] ; lpInBuffer
            mov     rcx, rdi        ; hDevice
            mov     edx, 53C028h    ; dwIoControlCode
            mov     r9d, 18h        ; nInBufferSize
            call    DeviceIoControl
            mov     esi, eax
            mov     rcx, rdi        ; hObject
            call    cs:__imp_CloseHandle
            mov     eax, esi
            add     rsp, 0E0h
            pop     rdi
            pop     rsi
            pop     rbp
            retn
```

Figure 1. Nokoyawa/Nevada code to delete Windows Shadow Copies

All versions of Nokoyawa support the command-line parameters *--file* (to encrypt a single file) and *--dir* (to encrypt a directory). However, Nokoyawa 1.1 and 2.0 require a configuration to execute the ransomware via the *--config* command-line parameter. The configuration parameter is a Base64 encoded JSON object that has the following keys and values shown in Table 2.

Key	Description
NOTE_NAME	Ransom note filename
NOTE_CONTENT	Ransom note content
EXTENSION	Encrypted file extension (also used as the Salsa20 nonce)
ECC_PUBLIC	Curve25519 public key
SKIP_EXTS	File extensions that will not be encrypted
SKIP_DIRS	Directories that will not be encrypted

ENCRYPT_NETWORK	Encrypt network shares
DELETE_SHADOW	Delete Windows shadow copies
LOAD_HIDDEN_DRIVES	Unhide hidden drives and encrypt files

Table 2. Nokoyawa 1.1 and Nokoyawa 2.0 ransomware configuration parameters

Nokoyawa 1.1 also has a *--safe-mode* command-line option to reboot the system into Windows safe mode prior to file encryption to maximize the number of files that can be encrypted by loading the minimal set of applications, and therefore, minimize the number of open file handles that may interfere with encryption. In addition, Nokoyawa 1.1 is the only variant that obfuscates the Windows API functions that are called during runtime by resolving each name via CRC32 hash.

In Nevada ransomware, the encryption parameters are hardcoded in the binary, but the other command-line options are virtually identical to Nokoyawa 1.1 and 2.0 (with the exception of a new feature to self-delete the ransomware binary after file encryption is complete). Nevada also supports a *-help* command-line argument, which prints the usage shown below in Figure 2.

```
How to run:
   nevada.exe -file <filePath> (encrypt selected file)
   nevada.exe -dir <dirPath> (encrypt selected directory)
   nevada.exe -sd (self delete after everything done)
   nevada.exe -sc (delete shadow copies)
   nevada.exe -lhd (load hidden drives)
   nevada.exe -nd (find and encrypt network shares)
   nevada.exe -sm (safe mode)                        © 2023 ThreatLabz
```

Figure 2. Nevada ransomware command-line help

In order to reduce the risk of law enforcement actions, Both Nokoyawa 2.0 and Nevada check whether the infected system is located in a former Commonwealth of Independent States (CIS) country. The former calls the Windows API *GetSystemDefaultLCID* for language IDs (between 1049-1092 or 2073) and the latter calls *GetUserDefaultUILanguage* (between 1049-1090) to determine the system's locale and language, respectively. Some of these language IDs include countries outside of the CIS countries, which may be to simplify the code by adding a range of values rather than individually checking each value.

Nokoyawa 1.0 and Nokoyawa 1.1 share about 39% of the same code, while Nokoyawa 2.0 and Nevada share more than 87% of the same code according to BinDiff.

Debug Print Statements

Another similarity between Nokoyawa 2.0 and Nevada are debug print statements, which are very similar or identical. Figure 3 shows an example for a function that creates a thread and prints a debug statement to the console.
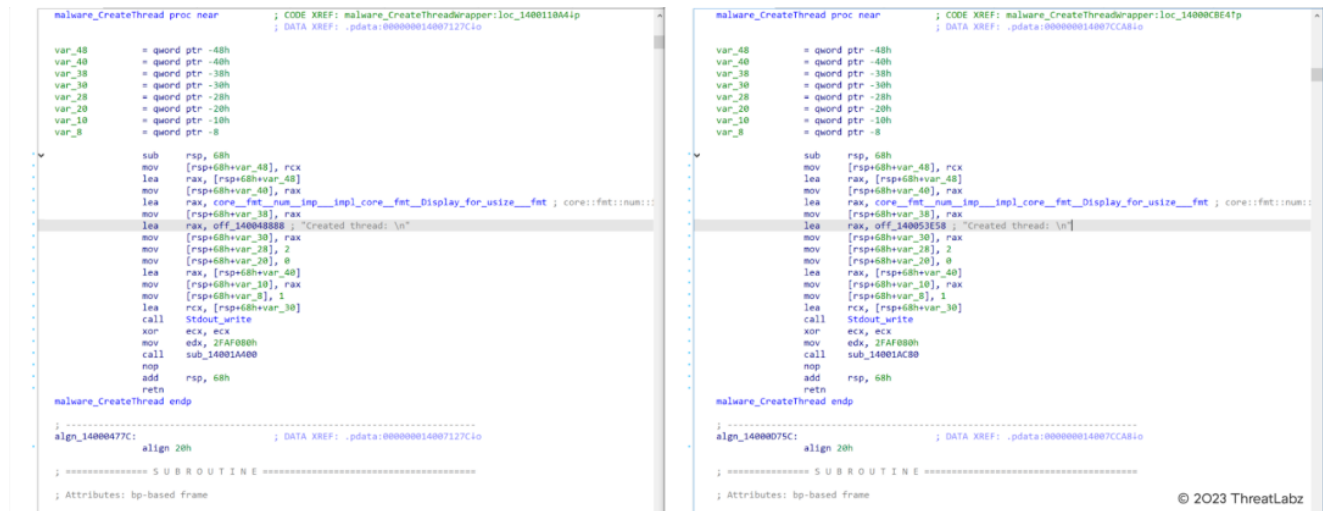

Figure 3. Comparison of *CreateThread* function and debug print statements in Nokoyawa 2.0 (left) and Nevada (right)

Many strings have also been slightly altered between Nokoyawa 2.0 and Nevada as shown in Table 3.

Nokoyawa 2.0	Nokoyawa 2.1 (Nevada)
CIS lang detected! Stop working...	CIS. STOP!
Successfully deleted shadow copies from	Shadow copies deleted from
Couldn't create ransom note	Failed to create ransom note
Couldn't seek file:	Failed to seek file:
Couldn't read file:	Failed to read file:
Couldn't write to file:	Failed to write file:
Couldn't rename file	Failed to rename file

Table 3. Comparison between debug print strings in Nokoyawa 2.0 (left) and Nevada (right)

Encryption Algorithms

Nokoyawa 1.0 and 1.1 use the elliptic curve SECT233R1 (NIST B-233) via the Tiny-ECDH library to generate a per file Salsa20 key. Nokoyawa 2.0 and Nevada use Curve25519 via the open source x25519_dalek Rust library to derive a Salsa20 encryption key per file. In Nokoyawa 1.1 and 2.0, the file extension (as described in Table 2) is used as the nonce. The original version of Nokoyawa and Nevada ransomware use the hardcoded nonce values *lvcelvce* and *pmarpmar*, respectively.

Conclusion

Zscaler ThreatLabz has identified two parallel versions of Nokoyawa ransomware with implementations in C and Rust. These two branches may be indicative of a source code leak, or designed to evade host-based security software and divert attention. In conclusion, Nevada ransomware appears to be the latest variant of the Rust-based version of Nokoyawa rather than an entirely new ransomware family.

Cloud Sandbox Detection

In addition to sandbox detections, Zscaler's multilayered cloud security platform detects indicators related to Nokoyawa at various levels with the following threat names:

Win64.Ransom.NOKOYAWA

Indicators Of Compromise (IOCs)

SHA256	Description
a32b7e40fc353fd2f13307d8bfe1c7c634c8c897b80e72a9872baa9a1da08c46	Nokoyawa ransomware 1.0
3339ba53e1f05f91dbe907d187489dbaba6c801f7af6fd06521f3ba8c484ec6c	Nokoyawa ransomware 1.1
7095beafff5837070a89407c1bf3c6acf8221ed786e0697f6c578d4c3de0efd6	Nokoyawa ransomware 2.0
855f411bd0667b650c4f2fd3c9fbb4fa9209cf40b0d655fa9304dcdd956e0808	Nokoyawa ransomware 2.1 (Nevada)

Thank you for reading

Was this post useful?

Yes, very!Not really

Get the latest Zscaler blog updates in your inbox

By submitting the form, you are agreeing to our privacy policy.