UNC5221: Unreported and Undetected WIREFIRE Web Shell Variant

By QuoINT

Published: 2024-01-22 · Archived: 2026-04-05 13:52:19 UTC

QuoIntelligence uncovers a previously unreported and undetected variant of the WIREFIRE web shell, a Python-based implant found in Ivanti Connect Secure (ICS) VPN compromised appliances.

Introduction

In mid-December, security researchers from Mandiant and Volexity identified multiple web shells hidden by an unknown threat actor on internal and external-facing web applications, a global exploitation and intrusion attempt against Ivanti Connect Secure (ICS) VPN appliances.

The threat actor, currently tracked by Mandiant as UNC5221 and by Volexity as UTA0178, appears to be part of a Chinese state-sponsored group with advanced capabilities in network exploitation of external-facing devices. Although there is a limited degree of certainty and public reporting in their attribution, indications suggest their activities are likely driven by espionage objectives.

During these incidents, the threat actor exploited two zero-day vulnerabilities (CVE-2023-21887 and CVE-2023-46805) affecting Ivanti Connect Secure VPN to bypass authentication methods and allowing the attackers to execute malicious commands on the targeted appliances. Additional information on the exploitation chain and methodologies used can be found in Volexity's initial article.

During our investigation, QuoIntelligence's Research Team discovered an additional variation of a Python web shell currently tracked as WIREFIRE (by Mandiant) and GIFTEDVISITOR (by Volexity) with similar capabilities, hidden in a different file. This discrepancy is presumably attributed to the threat actor's intent to circumvent detection mechanisms and to avoid detections by public YARA rules known.

To avoid confusion, the article will stick to the taxonomy provided by Mandiant.

At the time of reporting, public detections provided by security researchers are ineffective and will not detect this new variation, posing additional risks to compromised customers and clients that are undergoing internal investigations post-breach. The Integrity Check Tool provided by Ivanti will most likely detect a signature mismatch inside the application's folder, however, Volexity reported that UNC5221 was seen modifying the in-built scanner's code to prevent reporting any mismatch and further increase the likelihood of not detecting the tampered files.

Technical Details

During our research on Ivanti Connect Secure VPN exploits and hunts for additional samples, we identified a suspicious .EGG Python archive containing small variations between the original sample reported by Mandiant and Volexity.

In the original sample, WIREFIRE was located in the usual location /api/resources/visits.py and with the *post* function overwritten by the threat actor. However, our new finding was located inside of /api/resources/category.py, with the same *post* function overwritten.

The two code snippets highlighted minor differences in the methodology of data transmission and subsequent execution. POST requests with specific indicators remain the way to convey the encrypted data payload, which is then decrypted and directly executed within the memory space of the process, leaving no traces on the compromised file system.

In this particular instance, the threat actor can also rely on specifically set cookies to send the encrypted payload and will not rely on the GIF file previously seen in `visits.py`. If the cookie is not set, the malicious code will extract the contents of the API request containing the encrypted payload.

Our analysis highlights the emergence of a novel code addition that enables the threat actor to execute malicious code through the Python exec() function. This approach can facilitate the retention and persistence of data throughout successive POST requests, leveraging the globals() and locals() functions for data storage after each execution.

The following coding snippet highlights the content of the post function while redacting the value of variable *dskey*, containing a unique identifier of 16 characters used as the decryption key.

Figure1: Code snippet of the function “post” containing the slightly modified web shell

Detections

When the discovery was made, the existing YARA rule from Mandiant failed to identify our latest findings, leading us to the conclusion that is it likely that threat actors are deploying new web shells in different directories and with slight modifications to evade detections and hunting by security and IT teams using the publicly available YARA rules.

The rule M_Hunting_Dropper_WIREFIRE_1 that Mandiant provided detects the web shell WIREFIRE only if it resides inside the /api/resources/visits.py due to the strings used for matching, excluding any other web shell with identical or similar code but located in different files.

To rapidly respond to this new finding, we created a less restrictive and temporary YARA rule to detect commonalities between these web shells that are in different files, achieving to identify WIREFIRE and its variations in every file inside of /api/resources/ to test its accuracy.

```
{
meta:
```

```
  author = “QuoIntelligence”
  description = “Detects the web shell WIREFIRE tracked by Mandiant and similar variants using common pack / unpack methods”
   date = “2024-01-19”

 strings:

  $s1 = “zlib.decompress(aes.decrypt(base64.b64decode(” ascii
  $s2 = “from Cryptodome.Cipher import AES” ascii
   $p1 = “aes.encrypt(t+(‘\\x00’*(16-len(t)%16))” ascii

  condition:

  filesize < 10KB
  and all of ($s*)
   or any of ($p*)

}
```

Source: https://quointelligence.eu/2024/01/unc5221-unreported-and-undetected-wirefire-web-shell-variant/