# SockDetour – a Silent, Fileless, Socketless Backdoor – Targets U.S. Defense Contractors

unit42.paloaltonetworks.com/sockdetour

February 24, 2022

By Unit 42

February 24, 2022 at 6:00 AM

Category: Malware

Tags: APT, backdoor, CVE-2021-28799, CVE-2021-40539, CVE-2021-44077, TiltedTemple, Windows


This post is also available in: 日本語 (Japanese)

## Executive Summary

Unit 42 has been tracking an APT campaign we name TiltedTemple, which we first identified in connection with its use of the Zoho ManageEngine ADSelfService Plus vulnerability CVE-2021-40539 and ServiceDesk Plus vulnerability CVE-2021-44077. The threat actors involved use a variety of techniques to gain access to and persistence in compromised systems and have successfully compromised more than a dozen organizations across the technology, energy, healthcare, education, finance and defense industries. In conducting further analysis of this campaign, we identified another sophisticated tool being used to maintain persistence, which we call SockDetour.

A custom backdoor, SockDetour is designed to serve as a backup backdoor in case the primary one is removed. It is difficult to detect, since it operates filelessly and socketlessly on compromised Windows servers. One of the command and control (C2) infrastructures that the threat actor used for malware distribution for the TiltedTemple campaign hosted SockDetour along with other miscellaneous tools such as a memory dumping tool and several webshells. We are tracking SockDetour as one campaign within TiltedTemple, but cannot yet say definitively whether the activities stem from a single or multiple threat actors.

Based on Unit 42's telemetry data and the analysis of the collected samples, we believe the threat actor behind SockDetour has been focused on targeting U.S.-based defense contractors using the tools. Unit 42 has evidence of at least four defense contractors being targeted by this campaign, with a compromise of at least one contractor.

Unit 42 also believes it is possible that SockDetour has been in the wild since at least July 2019. We did not find any additional SockDetour samples on public repositories, meaning that the backdoor successfully stayed under the radar for a long time.

Full visualization of the techniques observed, relevant courses of action and indicators of compromise (IoCs) related to this report can be found in the Unit 42 ATOM viewer.

Palo Alto Networks customers are protected from the threats described in this blog by Cortex XDR and WildFire, and can use AutoFocus for tracking related entities. Additionally, the YARA rule we attached at the end of this blog post can be used to detect SockDetour in memory.

| | |
|---|---|
| Vulnerabilities Discussed | CVE-2021-40539, CVE-2021-44077, CVE-2021-28799 |
| Operating System Affected | Windows |
| Related Unit 42 Topics | TiltedTemple, APT, backdoors |

## Table of Contents

## Background on the TiltedTemple Campaign

TiltedTemple is the name Unit 42 gives to a campaign being conducted by an advanced persistent threat (APT) or APTs, leveraging a variety of initial access vectors, to compromise a diverse set of targets globally. Our initial publications on TiltedTemple focused on attacks that occurred through compromised ManageEngine ADSelfService Plus servers and through ManageEngine ServiceDesk Plus.

The TiltedTemple campaign has compromised organizations across the technology, energy, healthcare, education, finance and defense industries and conducted reconnaissance activities against these industries and others, including infrastructure associated with five U.S. states.

We found SockDetour hosted on infrastructure associated with TiltedTemple, though we have not yet determined whether this is the work of a single threat actor or several.

## SockDetour Targets US Defense Industry

While the TitledTemple campaign was initially identified as starting in August 2021, we have recently discovered evidence that SockDetour was delivered from an external FTP server to a U.S.-based defense contractor's internet-facing Windows server on July 27, 2021.

The FTP server also hosted other miscellaneous tools used by the threat actor, such as a memory dumping tool and ASP webshells.

After analyzing and tracking these indicators, we were able to discover that at least three other U.S.-based defense contractors were targeted by the same actor.

## SockDetour Hosted by Compromised Home and SOHO NAS Server

The FTP server that hosted SockDetour was a compromised Quality Network Appliance Provider (QNAP) small office and home office (SOHO) network-attached storage (NAS) server. The NAS server is known to have multiple vulnerabilities, including a remote code execution vulnerability, CVE-2021-28799. This vulnerability was leveraged by various ransomware families in massive infection campaigns in April 2021. We believe the threat actor behind SockDetour likely also leveraged these vulnerabilities to compromise the NAS server. In fact, the NAS server was already infected with QLocker from the previous ransomware campaigns.

## Analysis of SockDetour

SockDetour is a custom backdoor compiled in 64-bit PE file format. It is designed to serve as a backup backdoor in case the primary one is detected and removed. It works on Windows operating systems that are running services with listening TCP ports. It hijacks network connections made to the pre-existing network socket and establishes an encrypted C2 channel with the remote threat actor via the socket. Thus, SockDetour requires neither opening a listening port from which to receive a connection nor calling out to an external network to establish a remote C2 channel. This makes the backdoor more difficult to detect from both host and network level.

In order for SockDetour to hijack an existing process's socket, it needs to be injected into the process's memory. For this reason, the threat actor converted SockDetour into a shellcode using an open source shellcode generator called Donut framework, then used the PowerSploit memory injector to inject the shellcode into target processes. The samples we found contained hardcoded target processes' IDs, which means the threat actor manually chose injection target processes from compromised servers.

After SockDetour is injected into the target process, the backdoor leverages the Microsoft Detours library package, which is designed for the monitoring and instrumentation of API calls on Windows to hijack a network socket. Using the DetourAttach() function, it attaches a hook to the Winsock accept() function. With the hook in place, when new connections are made to the service port and the Winsock accept() API function is invoked, the call to the accept() function is re-routed to the malicious detour function defined in SockDetour.

Other non-C2 traffic is returned to the original service process to ensure the targeted service operates normally without interference.

With such implementation, SockDetour is able to operate filelessly and socketlessly in compromised Windows servers, and serves as a backup backdoor in case the primary backdoor is detected and removed by defenders.

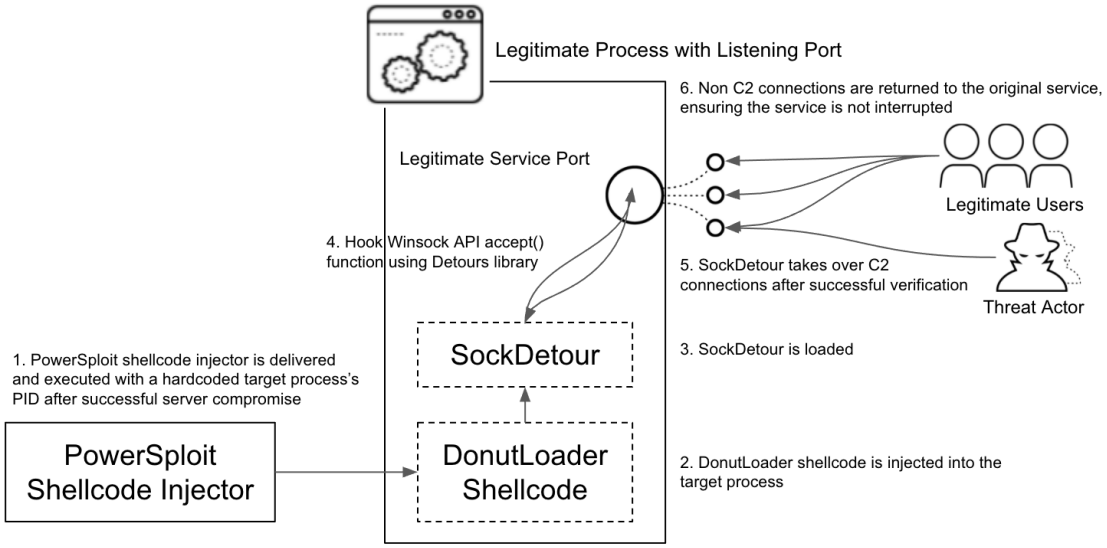


Figure 1. SockDetour Workflow

## Client Authentication and C2 Communication

As SockDetour hijacks all the connections made to the legitimate service port, it first needs to verify the C2 traffic from incoming traffic that is mixed with legitimate service traffic, then authenticate to make sure the C2 connection is made from the right client.

SockDetour achieves the verification and authentication of the C2 connection with the following steps.

1. First, expect to receive 137 bytes of data from a client for authentication. The authentication data is as shown in the structure in Table 1.

| *17 03 03* | *AA BB* | *CC DD EE FF* | *128-byte data block* |
|---|---|---|---|
| *Fixed header value to disguise TLS traffic* | *Payload data size* | *Four-byte variable used for client authentication* | *Data signature for client authentication data block* |

*Table 1. SockDetour client authentication data structure.*

2. Read the first nine bytes of data. This data is received using the recv() function with the MSG_PEEK option so that it will not interfere with the legitimate service's traffic by removing data from the socket queue.

3. Verify that the data starts with 17 03 03, which is commonly seen as a record header for TLS transactions when encrypted data is being transferred. However, this is abnormal for normal TLS – a TLS-encrypted transaction would not normally show up without proper TLS handshakes.

```
.text:000007FEFAB84823 ; ---------------------------------------------------------------------------
.text:000007FEFAB84823
.text:000007FEFAB84823 loc_7FEFAB84823:                        ; CODE XREF: _hookingFunc+87↑j
.text:000007FEFAB84823                  mov     r9d, MSG_PEEK   ; flags
.text:000007FEFAB84829                  mov     r8d, 9          ; len
.text:000007FEFAB8482F                  lea     rdx, [rsp+218h+_RecvBuf] ; buf
.text:000007FEFAB84837                  mov     rcx, [rsp+218h+s] ; s
.text:000007FEFAB8483C                  call    cs:recv         ; recv 9 bytes
.text:000007FEFAB84842                  mov     [rsp+218h+Buf2], 17h
.text:000007FEFAB84847                  mov     [rsp+218h+var_1D3], 3
.text:000007FEFAB8484C                  mov     [rsp+218h+var_1D2], 3
.text:000007FEFAB84851                  mov     r8d, 3          ; Size
.text:000007FEFAB84857                  lea     rdx, [rsp+218h+Buf2] ; Buf2
.text:000007FEFAB8485C                  lea     rcx, [rsp+218h+_RecvBuf] ; Buf1
.text:000007FEFAB84864                  call    memcmp          ; data should start with 17 03 03
.text:000007FEFAB84869                  test    eax, eax
```

Figure 2. SockDetour receives data with the MSG_PEEK option and verifies the data.

4. Check that the size of payload data AA BB is less than or equal to 251.

5. Check that the four bytes of payload CC DD EE FF satisfy the conditions below:

1. The result is 88 a0 90 82 after bitwise AND with 88 a0 90 82
2. The result is fd f5 fb ef after bitwise OR with fd f5 fb ef

6. Read the whole 137 bytes of data from the same data queue with the MSG_PEEK option for further authentication.

7. Build a 24-byte data block as shown in Table 2.

| *08 1c c1 78 d4 13 3a d7 0f ab* | *CC DD EE FF* | *b3 a2 b8 ae 63 bb 03 e8 ff 3b* |
|---|---|---|
| 10 bytes hardcoded in SockDetour | Four bytes received from the client for authentication | 10 bytes hardcoded in SockDetour |

*Table 2. 24-byte data block to be verified for client authentication.*

8. This 24-byte data block is hashed and verified using an embedded public key against the 128-byte data signature in Table 1, which the threat actor would have created by signing the hash of the same 24-byte data block using the corresponding private key.

This completes the client authentication step. After successful authentication, SockDetour takes over the TCP session using the recv() function without the MSG_PEEK option as this session is now verified to be for the backdoor.

Next, SockDetour creates a 160-bit session key using a hardcoded initial vector value bvyiafszmkjsmqgl, then sends it to the remote client using the following data structure.

| *17 03 03* | *AA BB* | *CC DD EE FF* | *session_key* | *random_padding* |
|---|---|---|---|---|
| *Fixed header value to disguise TLS traffic* | Payload data size | Session key length | 160-bit session key | Random padding |

*Table 3. SockDetour sending session key to client.*

In common encryption protocols such as TLS, the session key is encrypted with a public key before transferring. However, in this case, the malware author has seemingly forgotten the step and transfers the key in plain text.

Now with the session key shared between SockDetour and the remote client, the C2 connection is made encrypted over the hijacked socket.

## Plugin Loading Feature

As a backup backdoor, SockDetour serves only one feature of loading a plugin DLL. After the session key sharing, SockDetour receives four bytes of data from the client, which indicates the length of data SockDetour will receive for the final payload delivery stage. The size is expected to be smaller or equal to five MB.

The final payload data received is encrypted using the shared session key. After decryption, the received data is expected to be in JSON format with two objects app and args. app contains a base 64-encoded DLL, and args contains an argument to be passed to the DLL. SockDetour loads this plugin DLL in newly allocated memory space, then calls an export function with the name ThreadProc with a function argument in the following JSON structure.

```
1  {
2      "sock": hijacked_socket,
3      "key": session_key,
4      "args": arguments_received_from_client
5  }
```

While plugin DLL samples were not discovered, the above function argument suggests that the plugin also likely communicates via the hijacked socket and encrypts the transaction using the session key. Thus, we surmise it operates as stealthily as SockDetour does.

## Conclusion

SockDetour is a backdoor that is designed to remain stealthily on compromised Windows servers so that it can serve as a backup backdoor in case the primary one fails. It is filelessly loaded in legitimate service processes and uses legitimate processes' network sockets to establish its own encrypted C2 channel.

While it can be easily altered, the compilation timestamp of the SockDetour sample we analyzed suggests that it has likely been in the wild since at least July 2019 without any update to the PE file. Plus, we did not find any additional SockDetour samples on public repositories. This suggests that the backdoor successfully stayed under the radar for a long time.

The plugin DLL remains unknown, but it is also expected to operate very stealthily by being delivered via the SockDetour's encrypted channel, being loaded filelessly in memory and communicating via hijacked sockets.

As an additional note, the type of NAS server that we found hosting SockDetour is typically used by small businesses. This example serves as a critical reminder to patch this type of server frequently when fixes are released.

## Protections and Mitigations

Cortex XDR protects endpoints and accurately identifies the memory injector as malicious. Additionally, Cortex XDR has several detections for lateral movement and credential theft tactics, techniques and procedures (TTPs) employed by this actor set.

WildFire cloud-based threat analysis service accurately identifies the injector used in this campaign as malicious.

AutoFocus customers can track SockDetour activity via the SockDetour tag.

We advise server administrators to keep Windows servers up to date.

The YARA rule attached at the end of this blog can be used to detect the presence of SockDetour in memory.

Organizations should conduct an incident response investigation if they think they are compromised by SockDetour. If you think you may have been compromised or have an urgent matter, get in touch with the Unit 42 Incident Response team or call North America Toll-Free: 866.486.4842 (866.4.UNIT42), EMEA: +31.20.299.3130, APAC: +65.6983.8730 or Japan: +81.50.1790.0200.

## Indicators of Compromise

### SockDetour PE

0b2b9a2ac4bff81847b332af18a8e0705075166a137ab248e4d9b5cbd8b960df

### PowerSploit Memory Injectors Delivering SockDetour

80ed7984a42570d94cd1b6dcd89f95e3175a5c4247ac245c817928dd07fc9540
bee2fe0647d0ec9f2f0aa5f784b122aaeba0cddb39b08e3ea19dd4cdb90e53f9
a5b9ac1d0350341764f877f5c4249151981200df0769a38386f6b7c8ca6f9c7a
607a2ce7dc2252e9e582e757bbfa2f18e3f3864cb4267cd07129f4b9a241300b
11b2b719d6bffae3ab1e0f8191d70aa1bade7f599aeadb7358f722458a21b530
cd28c7a63f91a20ec4045cf40ff0f93b336565bd504c9534be857e971b4e80ee
ebe926f37e7188a6f0cc85744376cdc672e495607f85ba3cbee6980049951889
3ea2bf2a6b039071b890f03b5987d9135fe4c036fb77f477f1820c34b341644e
7e9cf2a2dd3edac92175a3eb1355c0f5f05f47b7798e206b470637c5303ac79f
bb48438e2ed47ab692d1754305df664cda6c518754ef9a58fb5fa8545f5bfb9b

### Public Key Embedded in SocketDetour

-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDWD9BUhQQZkagIIHsCdn/wtRNXcYoEi3Z4PhZkH3mar20EONVyXWP/YUxyUmxD+aT
----END PUBLIC KEY-----

### YARA Rule for Detecting SockDetour in Memory

```
rule apt_win_sockdetour
{
meta:
author = "Unit 42 - PaloAltoNetworks"
date = "2022-01-23"
description = "Detects SockDetour in memory or in PE format"
hash01 = "0b2b9a2ac4bff81847b332af18a8e0705075166a137ab248e4d9b5cbd8b960df"

strings:
$public_key =
"MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDWD9BUhQQZkagIIHsCdn/wtRNXcYoEi3Z4PhZkH3mar20EONVyXWP/YUxyUmxD
$json_name_sequence = {61 70 70 00 61 72 67 73 00 00 00 00 73 6F 63 6B 00 00 00 00 6B 65 79 00 61 72 67 73 00 00}
$verification_bytes = {88 [4] A0 [4] 90 [4] 82 [4] FD [4] F5 [4] FB [4] EF}
$data_block = {08 [4] 1C [4] C1 [4] 78 [4] D4 [4] 13 [4] 3A [4] D7 [4] 0F [4] AB [4] B3 [4] A2 [4] B8 [4] AE [4] 63 [4] BB [4] 03 [4] E8 [4] FF [4] 3
$initial_vector = {62 [4] 76 [4] 79 [4] 69 [4] 61 [4] 66 [4] 73 [4] 7A [4] 6D [4] 6B [4] 6A [4] 73 [4] 6D [4] 71 [4] 67 [4] 6C}

condition:
any of them
}
```


#### Get updates from Palo Alto Networks!

Sign up to receive the latest news, cyber threat intelligence and research from us

By submitting this form, you agree to our Terms of Use and acknowledge our Privacy Statement.