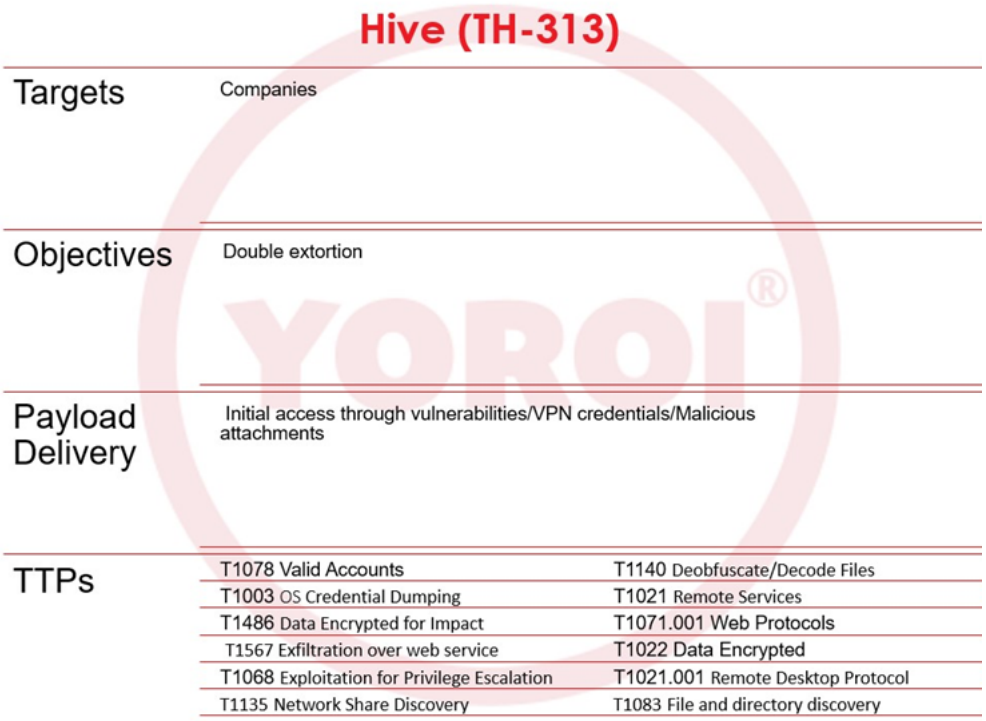
On the FootSteps of Hive Ransomware

yoroi.company/research/on-the-footsteps-of-hive-ransomware/

July 26, 2022



07/26/2022

Introduction

Hive ransomware is one of the most active financially motivated threat actors of this period, adopting the current Double Extorsion model. They started their malicious activities in June of the past year, and just in a year of activity they collected a big number of victims, demonstrating the capability to hit even critical infrastructures.

The criminal group distinguished from other ones also for attacking healthcare organization during the 2021 when we had to face off the Covid-19 pandemic. It was emblematic that one of the first victims was the Memorial Health System in August 2021.

For these reasons, Yoroi's Malware ZLab decided to keep track of this infamous threat actor and observe any modification of its modus operandi, in order to provide a guideline focusing on the evolution of the locker sample of the cyber gang.

About Hive

Hive (TH-313) is a Ransomware group firstly spotted in June 2021 and it gathered a big popularity inside the cybersecurity community because it was able to attack a large variety of sectors, starting from healthcare facilities and arriving to critical infrastructures, passing through manufacturers during just a year of activity.

In addition, the group was able to refine its toolkit and then its TTPs with a surprising speed: the business model is the Double-Extorsion and Ransomware-as-a-Service, with a self-made ransomware payload.

Hive (TH-313)

Targets	Companies
Objectives	Double extortion
Payload Delivery	Initial access through vulnerabilities/VPN credentials/Malicious attachments

TTPs	
T1078 Valid Accounts	T1140 Deobfuscate/Decode Files
T1003 OS Credential Dumping	T1021 Remote Services
T1486 Data Encrypted for Impact	T1071.001 Web Protocols
T1567 Exfiltration over web service	T1022 Data Encrypted
T1068 Exploitation for Privilege Escalation	T1021.001 Remote Desktop Protocol
T1135 Network Share Discovery	T1083 File and directory discovery

Figure 1: Hive (TH-313)

So, in this report we have decided to focus our attention on the ransomware payload evolution, providing a timeline of the development of Hive Ransomware Payloads.

Timeline of the development of Hive Ransomware

Inside the criminal group, there is surely a high-profile development team, with deep knowledge of programming in both newer and older programming languages. The first versions of the encryptor payload are written in Golang, then, starting from the v5 version, the dev team of Hive switched into Rust.

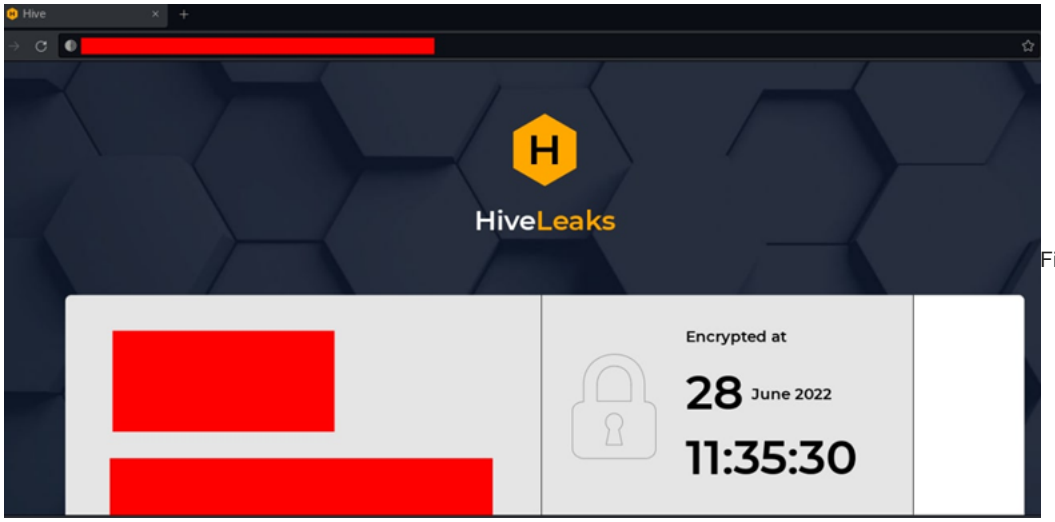


Figure 2: Leak Site

In the following timeline, we provide a quick overview of the evolution of the malware and how the cyber gang adopted an incremental development process on its TTPs:

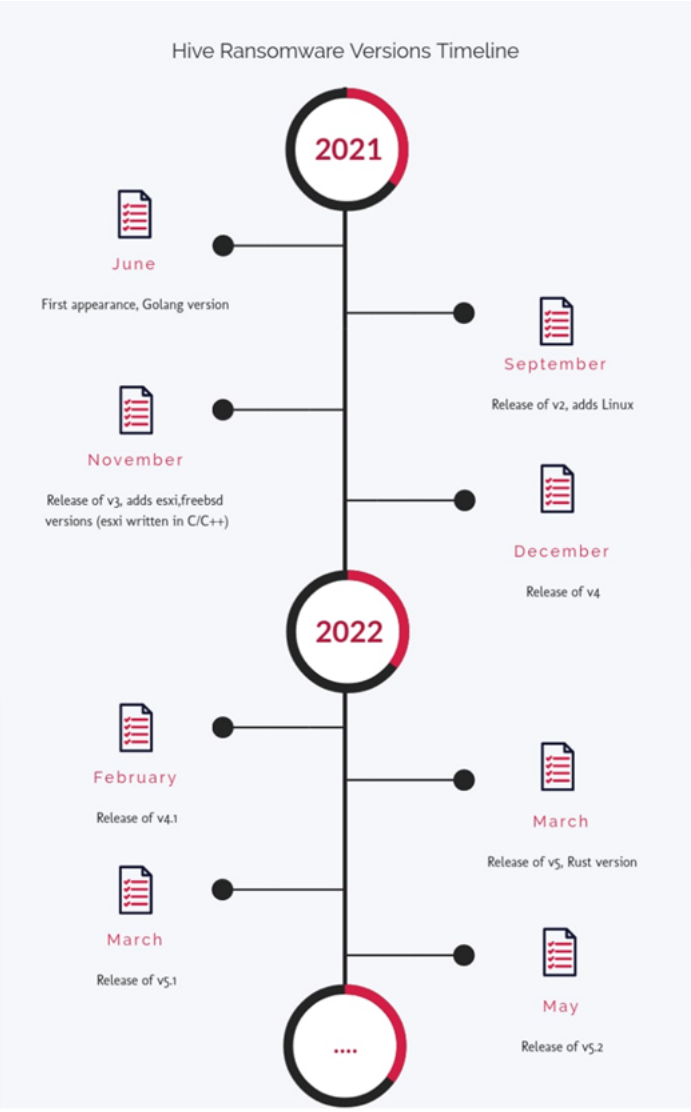


Figure 3: Hive Timeline

In the same way, even the Ransom Note changed during the evolution: first, the credentials were hardcoded inside the sample, but now the operators pass them as a parameter when the locker process is launched. Below a comparison between an earlier version and a later one:

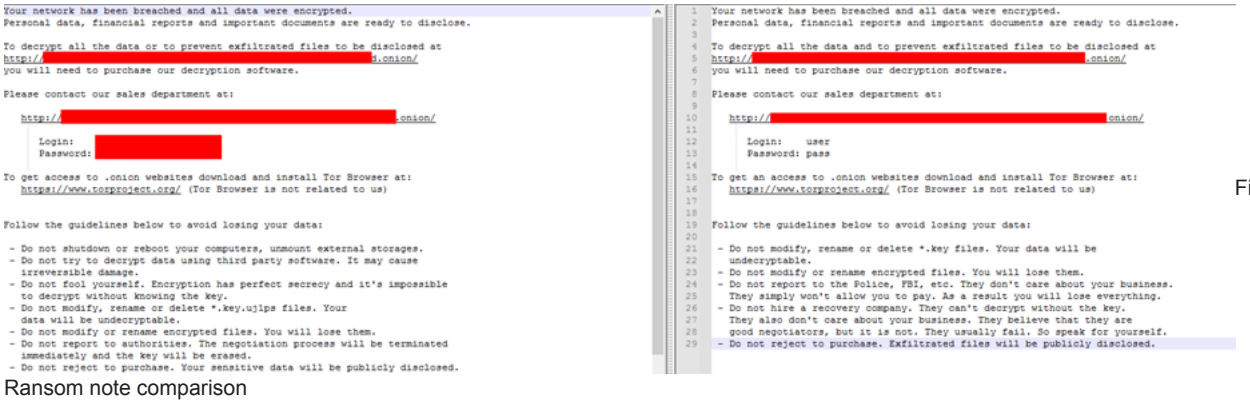


Figure 4: Ransom note comparison

Victimology

During its activity, Hive Group hit a large number of victims and during that period some of them paid the ransom, after that the victims were removed from the “walk of shame”. We tracked a total of 130 victims listed on their leak site, the affected companies belong to different sectors and nations. However, we have evidence that occasionally some victims of the group, despite being attacked by the threat actor, are never reported onto the site.

Moreover, the group does not exclude hospitals, companies that provide medical equipment and non-profit organizations. An example is the attack on the "Memorial Health System" in August 2021 or more recently on the "Partnership HealthPlan of California", a non-profit organization.

The following graph shows the total progress of the victims so far, indicating that the group is consolidating its role as one of the principal threats in the panorama.

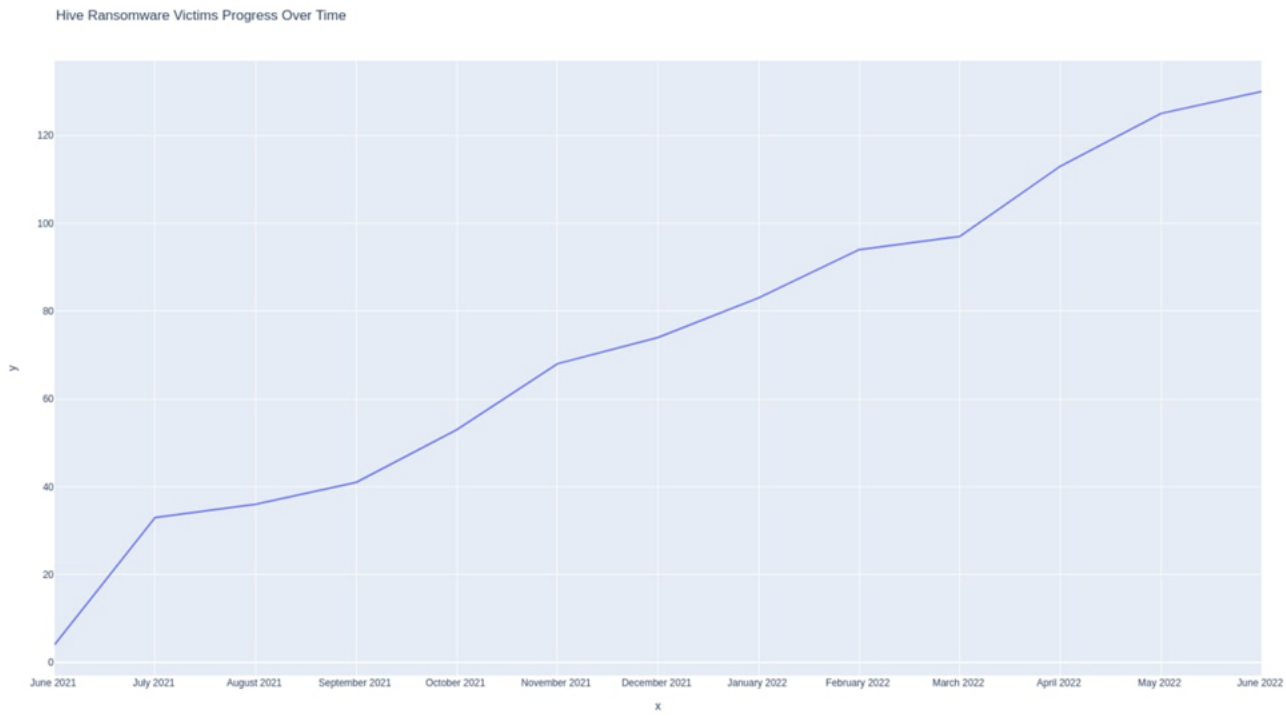


Figure 5: Hive Ransomware Victims Progress over time

Another view of the same information is represented in the following graph, where the focus is pointed to highlight the month in which most victims were published on their leak site, which turns out to be July 2021, shortly after the group started. So, it means that the ransomware operators gathered a consistent number of victims during the startup phase, in order to create a solid placement inside the threat landscape. After that phase, the gang continued to threaten with huge aggression.

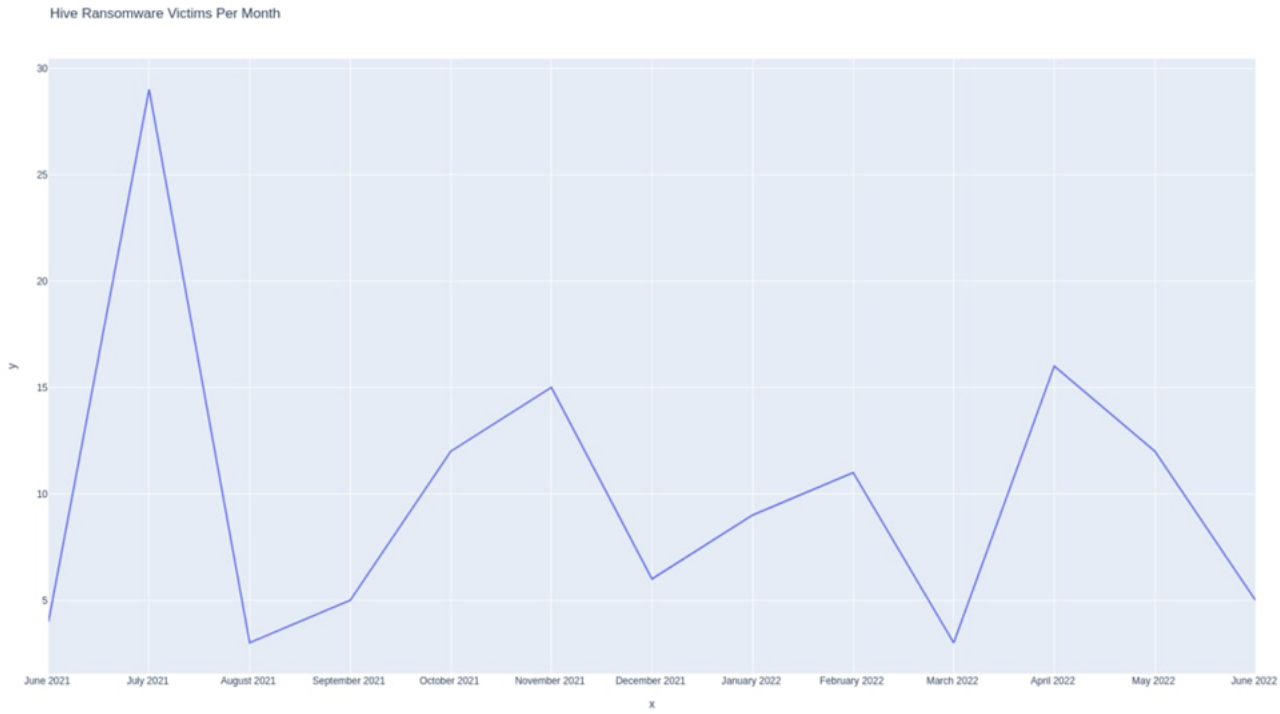


Figure 6: Hive Victims Per Month

Hive v1

Hash	88f7544a29a2ceb175a135d9fa221cbfd3e8c71f32dd6b09399717f85ea9afd1
Threat	Ransomware
Brief Description	Hive Ransomware v1
SSDEEP	12288:CinNFNkY/yU97ppM4NSBG81Np2C9H4S3iDjlLtc4wCIITIQaOI6NrwacVYV+4MsT:CinN3n/y67jM4v4kCSPDjlLtbwt8IQLH

Table 1: Hive v1

The first version, written in Golang, was a sophisticated encryptor program, but, due to the newness of the malicious activity, there is no track of obfuscation, and the strings can be easily seen, the following figure shows some of the available parameters:

```
lea     eax, aNumberOfEncryp ; "Number of encryptor threads"
mov     [esp+7Ch+var_6C], eax
mov     [esp+7Ch+var_68], 1Bh
call    flag___ptr_FlagSet__Int
nop
mov     eax, [esp+7Ch+var_64]
mov     [esp+7Ch+var_48], eax
mov     ecx, dword_61DE10
mov     [esp+7Ch+var_7C], ecx
lea     ecx, aStop      ; "stop"
mov     [esp+7Ch+var_78], ecx
mov     [esp+7Ch+var_74], 4
lea     ecx, aBmrSqlOraclePo ; "bmr|sql|oracle|postgres|redis|vss|backu"...
mov     [esp+7Ch+var_70], ecx
mov     [esp+7Ch+var_6C], 2Dh ; '-'
lea     ecx, aRegexpToMatchS ; "Regexp to match services to stop, case "...
mov     [esp+7Ch+var_68], ecx
mov     [esp+7Ch+var_64], 32h ; '2'
call    flag___ptr_FlagSet__String
nop
mov     eax, [esp+7Ch+var_60]
mov     [esp+7Ch+var_44], eax
mov     ecx, dword_61DE10
mov     [esp+7Ch+var_7C], ecx
lea     ecx, aKill      ; "kill"
mov     [esp+7Ch+var_78], ecx
mov     [esp+7Ch+var_74], 4
lea     ecx, aMspubMsdesktop ; "mspub|msdesktop"
mov     [esp+7Ch+var_70], ecx
mov     [esp+7Ch+var_6C], 0Fh
lea     ecx, aRegexpToMatchN ; "Regexp to match names of processes to k"...
mov     [esp+7Ch+var_68], ecx
mov     [esp+7Ch+var_64], 3Ch ; '<'
call    flag___ptr_FlagSet__String
```

Figure 7: Available parameters

The initial effort of the gang was to make a product quite customizable according to the infection and the encryption process to perform. In this way, the malware writers provided a series of parameters to launch an ad-hoc infection profile.

The following table describes all the available parameters found in this version:

Parameter	Description
-kill	Regex, names of the processes to kill. Default values: “mspub|msdesktop”
-no-clean	Skip clean disk space stage
-skip	Regex, names of the files to skip. Default values: “\\.lnk”
-skip-before	Skip files before a specific date. Defaut value: “03.09.2016”
-stop	Regex, names of the services to stop. Default values: "bmr|sql|oracle|postgres|redis|vss|backup|sstp"
-t	Number of threads

Table 2: Hive v1 Parameters

Once the parameters are parsed, creating the desired infection profile, the control flow passes to the core malicious operations.

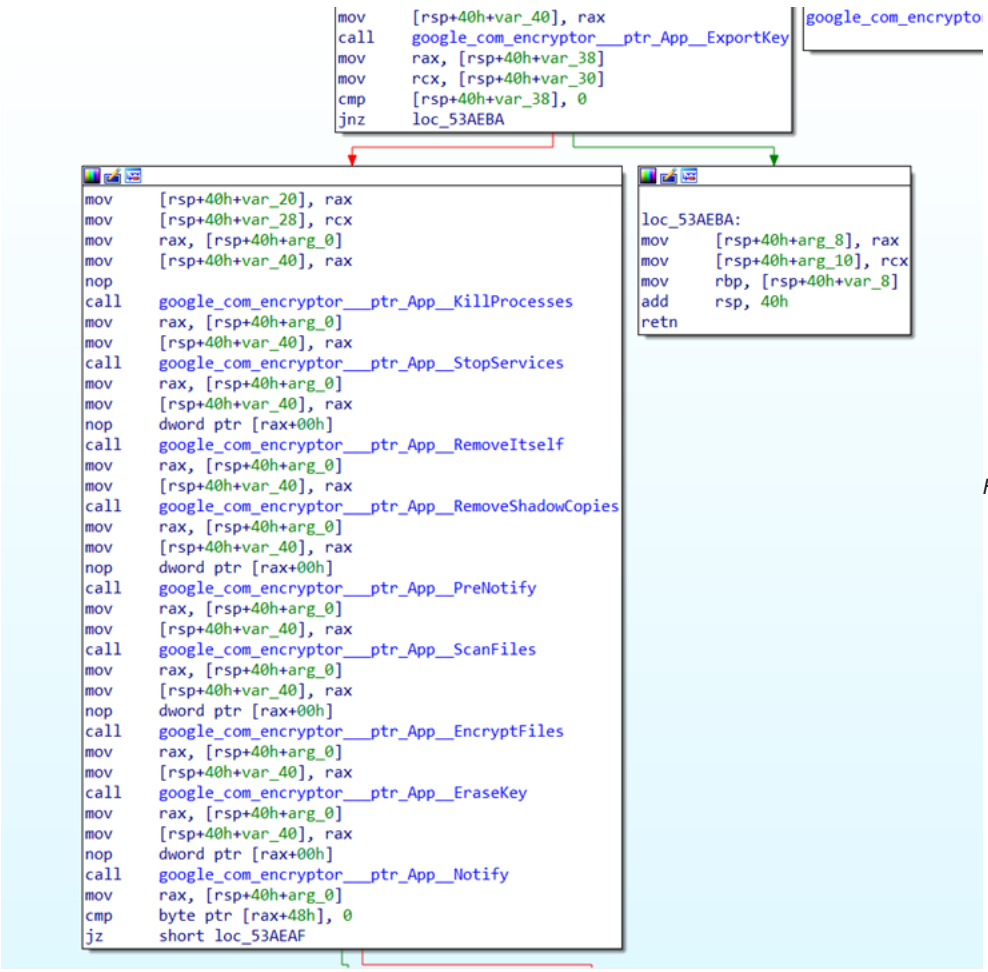


Figure 8: Hive core function

The locker sample proceeds to export the key, to kill the processes and services specified and to remove the shadow copies then it iterates the directories and starts encrypting the files.

The core of the encryption scheme of Hive ransomware is a union of XOR+RSA algorithms. In the figure below we can see the XOR related routine:

```
    v70[v66] ^= *(_BYTE *)(v24 + v71) ^ *(_BYTE *)(v23 + v72);
    v19 = v56;
    v66 = (unsigned int)v66 + 1LL;
    v21 = HIDWORD(v64);
    v20 = v64;
}
if ( v56 > 0x1000 )
    runtime_panicSliceAcap(v37, v41);
v67 = v22;
v56 = os___ptr_File__WriteAt(v73, v70, v56, 4096, v20, v21);
```

Figure 9: Usage of XOR algorithm

Then, in this first version, it uses ".hive" as extension to the encrypted files, later it is used a unique ID instead. Moreover, the **RemoveItself routine** drops "hive.bat" to remove itself. But, since the second version of the malware calls the related function after the encryption is complete:

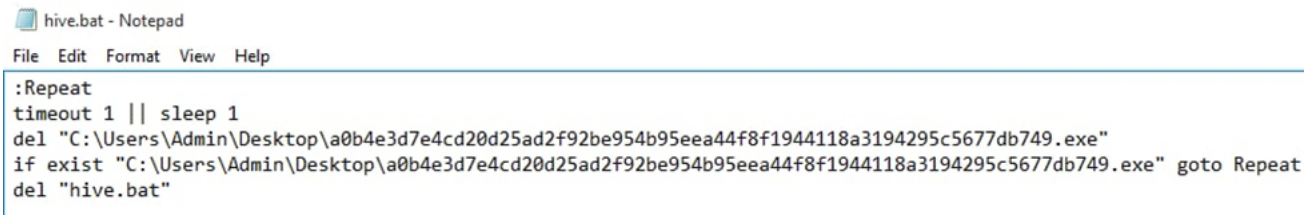


Figure 10: hive.bat

RemoveShadowCopies drops "shadow.bat" to remove the shadow copies, from the second version will directly execute the command instead of dropping a .bat:

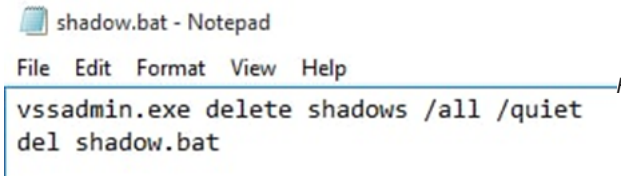


Figure 11: shadow.bat

Hive V2

Hash	25bfec0c3c81ab55cf85a57367c14cc6803a03e2e9b4afd72e7bbca9420fe7c5
Threat	Ransomware
Brief Description	Hive Ransomware v2
SSDEEP	12288:Sw41dVZvThPCsM18GLHe7wlDdkPAQEtxr0fflvRmhEBWtdUJiAUtP/T/kAfMvgV:dod1HDmlDdkZ4YXPpaTTXMw

Table 3: Hive v2

With the second version of Hive, the malware writers started to complicate the code in order to make the analysis more difficult for the analyst. The initial step is to obfuscate the “Go Build ID” header present in all golang- written binaries.

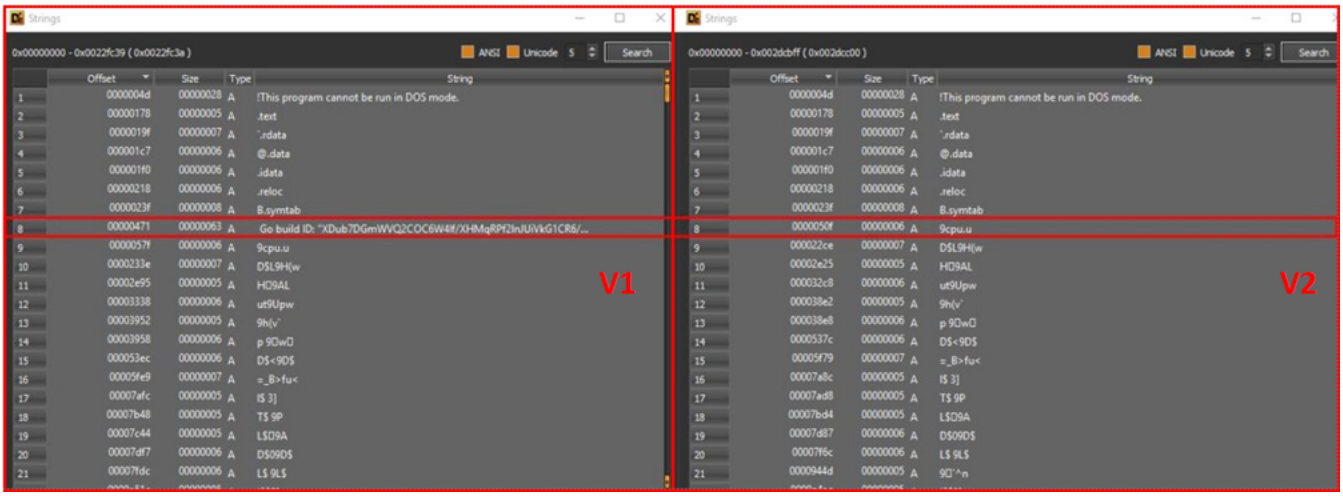


Figure 12: Strings comparison

The simple trick causes that, when opening a disassembler, like IDA, the analyst can immediately see Golang not being recognized. However, a simple fix provides the overwriting of the build-id with a legit one.

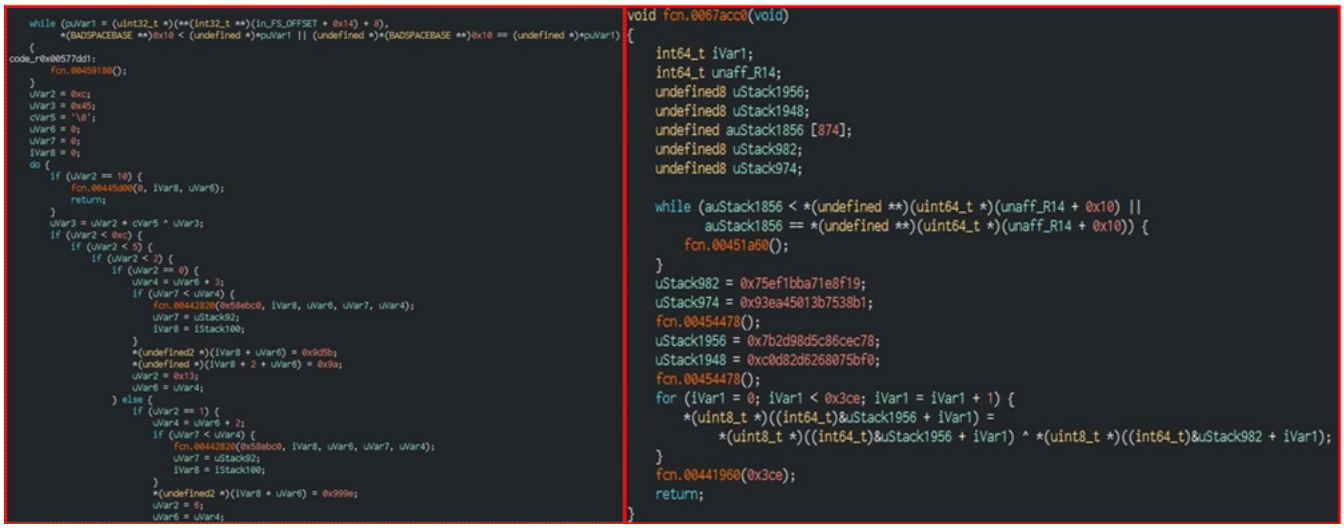
Figure 13: “Go Build ID” patch

In addition, now the strings are obfuscated, and the names of the functions present inside the main are not visible in cleartext:

```
mov     [esp+70h+var_5C], ecx
mov     [esp+70h+var_58], edx
call    flag___ptr_FlagSet__String
mov     eax, [esp+70h+var_54]
mov     [esp+70h+var_28], eax
call    main_main_func4
mov     eax, [esp+70h+var_70]
mov     [esp+70h+var_1C], eax
mov     ecx, [esp+70h+var_6C]
mov     [esp+70h+var_3C], ecx
call    main_main_func5
mov     eax, [esp+70h+var_70]
mov     [esp+70h+var_2C], eax
mov     ecx, [esp+70h+var_6C]
mov     [esp+70h+var_4C], ecx
call    main_main_func6
nop
mov     eax, dword_6C7F10
mov     ecx, [esp+70h+var_70]
mov     edx, [esp+70h+var_6C]
mov     [esp+70h+var_70], eax
mov     eax, [esp+70h+var_1C]
mov     [esp+70h+var_6C], eax
mov     eax, [esp+70h+var_3C]
mov     [esp+70h+var_68], eax
mov     eax, [esp+70h+var_2C]
mov     [esp+70h+var_64], eax
mov     eax, [esp+70h+var_4C]
mov     [esp+70h+var_60], eax
mov     [esp+70h+var_5C], ecx
mov     [esp+70h+var_58], edx
call    flag___ptr_FlagSet__String
mov     eax, [esp+70h+var_54]
mov     [esp+70h+var_10], eax
call    main_main_func7
mov     eax, [esp+70h+var_70]
```

Figure 14: Obfuscated parameters

In the following screen two different routines for the strings obfuscation is provided:



```
void fcn.0067acc0(void)
{
    int64_t iVar1;
    int64_t unaff_R14;
    undefined8 uStack1956;
    undefined8 uStack1948;
    undefined auStack1856 [874];
    undefined8 uStack982;
    undefined8 uStack974;

    while (auStack1856 < *(undefined **)(uint64_t *)(unaff_R14 + 0x10) ||
           auStack1856 == *(undefined **)(uint64_t *)(unaff_R14 + 0x10)) {
        fcn.00451a60();
    }
    uStack982 = 0x75ef1bba71e8f19;
    uStack974 = 0x93ea45013b7538b1;
    fcn.00454478();
    uStack1956 = 0x7b2d98d5c86cec78;
    uStack1948 = 0xc0d82d6268075bf0;
    fcn.00454478();
    for (iVar1 = 0; iVar1 < 0x3ce; iVar1 = iVar1 + 1) {
        *(uint8_t *)((int64_t)&uStack1956 + iVar1) =
             *(uint8_t *)((int64_t)&uStack1956 + iVar1) ^ *(uint8_t *)((int64_t)&uStack982 + iVar1);
    }
    fcn.00441960(0x3ce);
    return;
}
```

Figure 15: Strings decryption routines

The help command has also changed, it has more default values, the “-t” and “-skip” parameters have been removed, “-grant” has been added and “-no-clean” renamed to “-no-wipe”

Parameter	Description
-grant	Grant permissions to all files
-kill	Regex, names of the processes to kill. Default values: "agntsvc|sql|CNTAoSMgr|dbeng50|dbsnmp|encsvc|excel|firefoxconfig|infopat
-no-wipe	Skip wipe of free space

-stop	Regex, names of the services to stop. Default values: "acronis|AcrSch2Svc|Antivirus|ARSM|AVP|backup|bedbg|CAARCUpdateSvc| |mfemms|mfevtp|MMS|MsDtsServer|MsDtsServer100|MsDtsServer110|msexchange|msmdsrv|MSOLAP|MVArmor|MVarmor64|NetM

Table 4: Hive v2 parameters

The string obfuscation process does not impact the structure of the main function, following a comparison of these two versions.

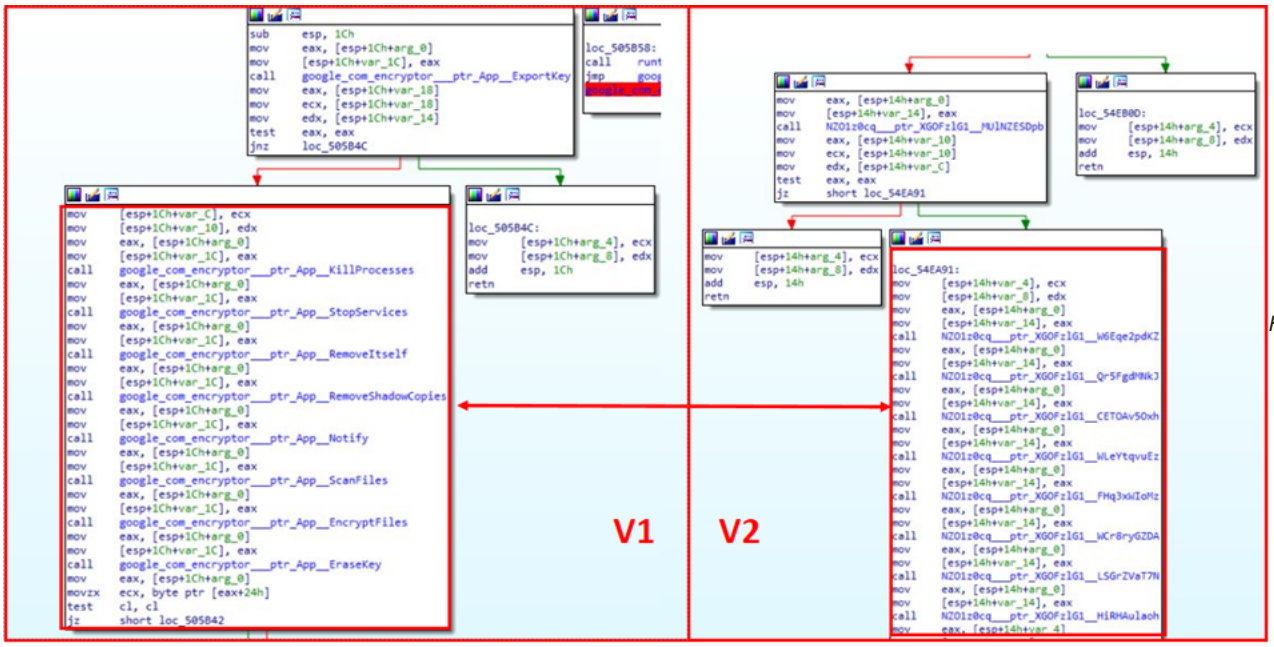


Figure 16: Functions name comparison

Hive V3

Hash	8a461e66ae8a53ffe98d1e2e1dc52d015c11d67bd9ed09eb4be2124efd73ccd5
Threat	Ransomware
Brief Description	Hive Ransomware v3
SSDEEP	49152:gWVNVvSGbjmrb/T6vO90dL3BmAFd4A64nsfJuhQ8jmp4S3C5CEg+eNgiQJfOqAD:gWYQjPhQCmppnMfO

Table 5: Hive v3

In this version the “-skip” parameter has been restored and, in another sample (3858e95bcf18c692f8321e3f8380c39684edb90bb622f37911144950602cea21), we found a new parameter named “scan”:

Parameter	Description
-scan	Scan local network for shares

Table 6: Hive v3 Additional parameters

Comparing the logs from the v1 we can spot the following differences:

- The key name is longer and it has a random extension
- It shows the time elapsed for the encryption of each file

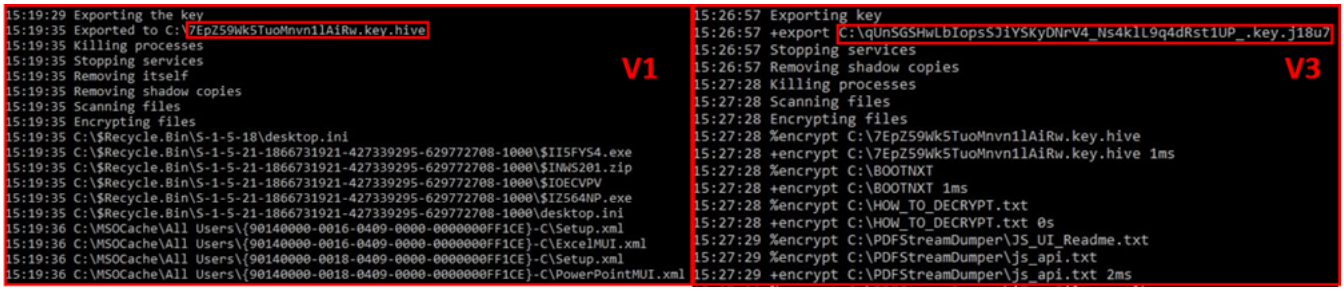


Figure 17: Comparison of logs

Linux/FreeBSD version

The third version of the development of Hive ransomware saw the porting of the codebase for other operating systems, such as Linux/FreeBSD and ESXi.

The Linux (12389b8af28307fd09fe080fd89802b4e616ed4c961f464f95fdb4b3f0aaf185) and FreeBSD (bdf3d5f4f1b7c90dfc526340e917da9e188f04238e772049b2a97b4f88f711e3) versions are almost identical to the Windows one, despite the obvious OS differences. One of those differences is the following function “KillNonRoot” aimed at killing all non-root processes:

```
while ( (unsigned __int64)&v11 + 2 <= *(_QWORD *)(v0 + 16) )
  runtime_morestack_noctxt();
v15 = 0LL;
SsLVP2b0___ptr_LUvzP8mV__KillNonRoot_func1();
v14 = v1;
v6 = runtime_convTstring();
*(_QWORD *)&v14 = &unk_54DA20;
*((_QWORD *)&v14 + 1) = v2;
log_Println();
v11 = 0xC706374314BA012CLL;
v12 = -28055;
v9 = runtime_growslice(v6);
*v3 = 47;
v3[1] = 112;
v3[2] = 114;
v3[3] = 111;
v3[4] = 99;
v4 = v3;
runtime_slicebytetostring(v7, v9);
result = os_OpenFile();
if ( !v4 )
{
  v13[0] = sub_52B300;
  v13[1] = result;
  v15 = (__int64 (**)(void))v13;
  os___ptr_File__Readdir(v8, v10);
  return (*v15)();
}
return result;
}
```

Figure 18: “KillNonRoot” Function

Hive v3 ESXI

Hash	822d89e7917d41a90f5f65bee75cad31fe13995e43f47ea9ea536862884efc25
Threat	Ransomware
Brief Description	Hive Ransomware v3
SSDEEP	3072:3Zp7gZzdfvjRCMj1Yk36ioyJ1zgjIlOhXYopNL+V7o0xvvkB/37Nt7xhew8A2Mz:P7gDj8S1Hlx14+opNCIvk977ew8A2M

Table 7: Hive v3 ESXI

In this case, the malware is written in C/C++, in order to have a better compatibility with the target operating system, the strings are not obfuscated, and we have found some new parameters:

Parameter	Description
-no-stop	Don’t stop virtual machines
-low-cpu	Single thread encryption

Table 8: Hive v3 ESXI Parameters

After the routine of exporting the keys already seen in the previous paragraphs, the sample stops all the running virtual machines in order to encrypt them without problems:

```
int __fastcall sub_519E(__int64 a1)
{
  int result; // eax

  puts("Preprocess");
  sub_51EF(a1);
  result = *(unsigned __int8 *)(a1 + 66);
  if ( (_BYTE)result != 1 )
  {
    puts("Stopping VMs");
    return system("vim-cmd vmsvc/getallvms | grep -o -E '^[0-9]+' | xargs -r -n 1 vim-cmd vmsvc/power.off");
  }
  return result;
}
```

Figure 19: "Stopping virtual machines

The ransom note contains also a reference to not delete or reinstall the virtual machines:

```
"   %s\n"
"  \n"
"      Login:    %s\n"
"      Password: %s\n"
"\n"
"To get an access to .onion websites download and install Tor Browser at:\n"
"   https://www.torproject.org/ (Tor Browser is not related to us)\n"
"\n"
"\n"
"Follow the guidelines below to avoid losing your data:\n"
"\n"
" - Do not delete or reinstall VMs. There will be nothing to decrypt.\n"
" - Do not modify, rename or delete *.key.%s files. Your data will be \n"
"   undecryptable.\n"
" - Do not modify or rename encrypted files. You will lose them.\n"
" - Do not report to the Police, FBI, etc. They don't care about your business.\n"
"   They simply won't allow you to pay. As a result you will lose everything.\n"
" - Do not hire a recovery company. They can't decrypt without the key. \n"
"   They also don't care about your business. They believe that they are \n"
"   good negotiators, but it is not. They usually fail. So speak for yourself.\n"
" - Do not reject to purchase. Exfiltrated files will be publicly disclosed.\n",
```

Figure 20: Ransom note

As said, the objective of this version is to encrypt the virtual machines hosted on the ESXi server, so, the malware goes to find the virtual machines deployed on the server, by using a custom regex aimed at finding the words "vm" or "vs".

```
regcomp(*(regex_t **)(a1 + 72), "\\.(vm|vs)\\w+$", 1);
*(_QWORD *)(a1 + 80) = malloc(0x40uLL);
regcomp(*(regex_t **)(a1 + 80), "^$", 1);
v2 = sub_46B9();
snprintf(s, 0xFFuLL, "(.+)\\.(.+?)\\.%s$", v2);
*(_QWORD *)(a1 + 88) = malloc(0x40uLL);
regcomp(*(regex_t **)(a1 + 88), s, 1);
```

Figure 21: Regex for ESXI version

Hive v4

Hash	33aceb3dc0681a56226d4cfce32eee7a431e66f5c746a4d6dc7506a72b317277
Threat	Ransomware
Brief Description	Hive Ransomware v4
SSDEEP	49152:e2NiZPNNirb/T2vO90dL3BmAFd4A64nsfJk0NuXCdmTQb0/6VCrrPrsbg11VgWA:e2ANB04yla0hsirubO

Table 9: Hive v4

The fourth version of Hive locker is an effort to obfuscate also the code. We haven't noticed new features or upgrades except for a more serious obfuscation of the code and changes in the details of the key generation and encryption.

In detail, this version adopts the control flow flattering obfuscation technique, which is largely adopted by many attackers, thanks to its actual effectiveness. Below an example of that technique:

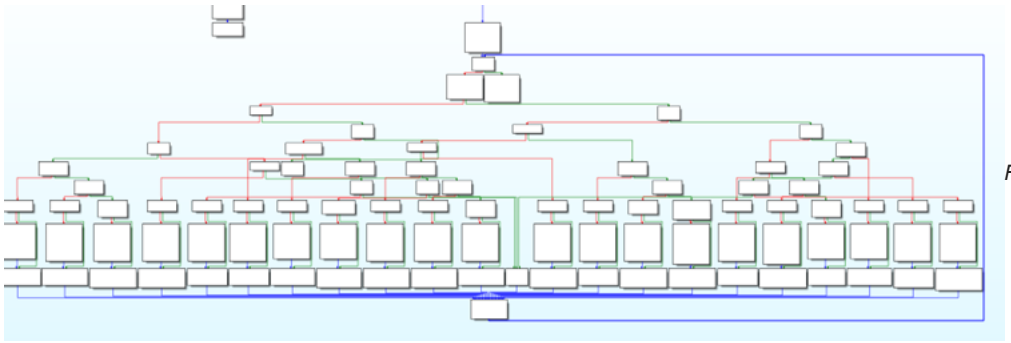


Figure 22: Control flattening obfuscation

Hive v5

The fifth version of hive represents a sort of revolution inside the entire codebase. In this version, the major differences include the changing of the base programming language and the refinement encryption algorithm.

Hash	b6b1ea26464c92c3d25956815c301caf6fa0da9723a2ef847e2bb9cd11563d8b
Threat	Ransomware
Brief Description	Hive Ransomware v5.2
SSDEEP	12288:BLF6OtM1z8JLbA689tSfvTvFSYIzp4yzhrWbttQfaa4GxjzgdIo/AhwN/eh9z/E:BLF6gb0xqx9z/EO3BxhR

Table 10: Hive v5

Hive is now written in Rust and for this reason the difficulty has increased, along with a complex encryption scheme makes the analysis harder even for experienced analysts.

The refinement of the encryption process considers the passing from "***XOR+RSA***" of the previous versions, arriving to "***ECDH+Curve25519+XChaCha20-Poly1305***"

For this version we found the following parameters:

Parameter	Description
-no-local	Don't encrypt local files
-no-mounted	Don't encrypt on mounted network volumes
-no-discovery	Don't discover network volumes
-local-only	Encrypt only local files
-network-only	Encrypt only network volumes
-explicit-only	Encrypt specified folders
-min-size	Minimum file size
-timerze-only	N/A
-da	N/A

Table 11: Hive v5 parameters

Once executed, the sample checks for the parameter "-u", which should contain the "*username:password*" used as credentials for the victim and written in the ransom note, if this unique parameter is missing, the program exits.

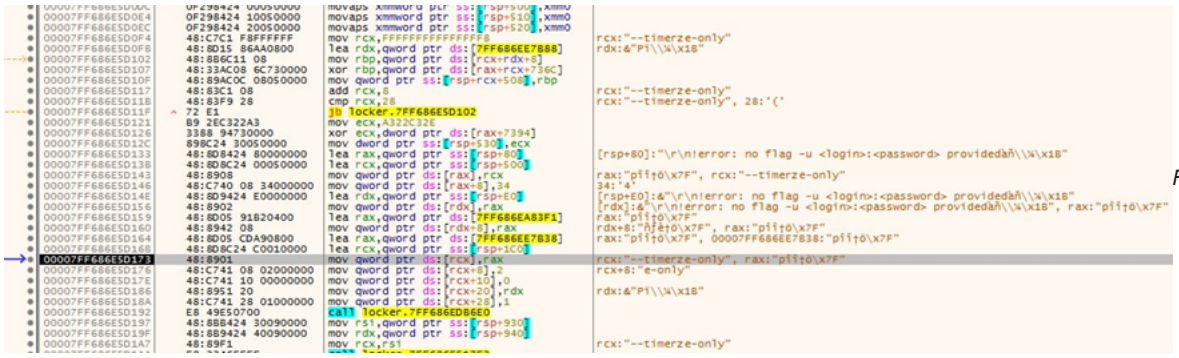


Figure 23: -u parameter

Even the routine to decrypt the ransom note changed. In this case, the protection of the ransom note relies on a XOR key.

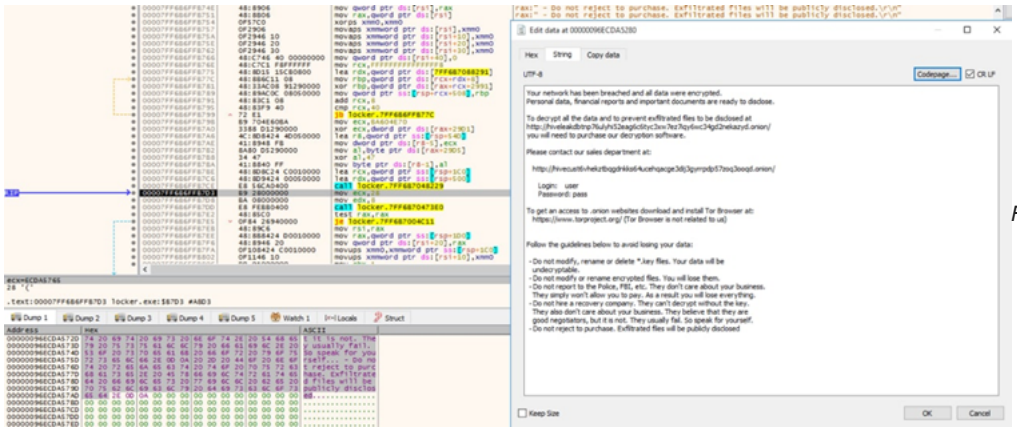


Figure 24: Decrypted Ransom Note

Another update is the expansion on the other drives. The sample generates an array of drive labels and uses **GetDriveTypeW** to check if the path is invalid:

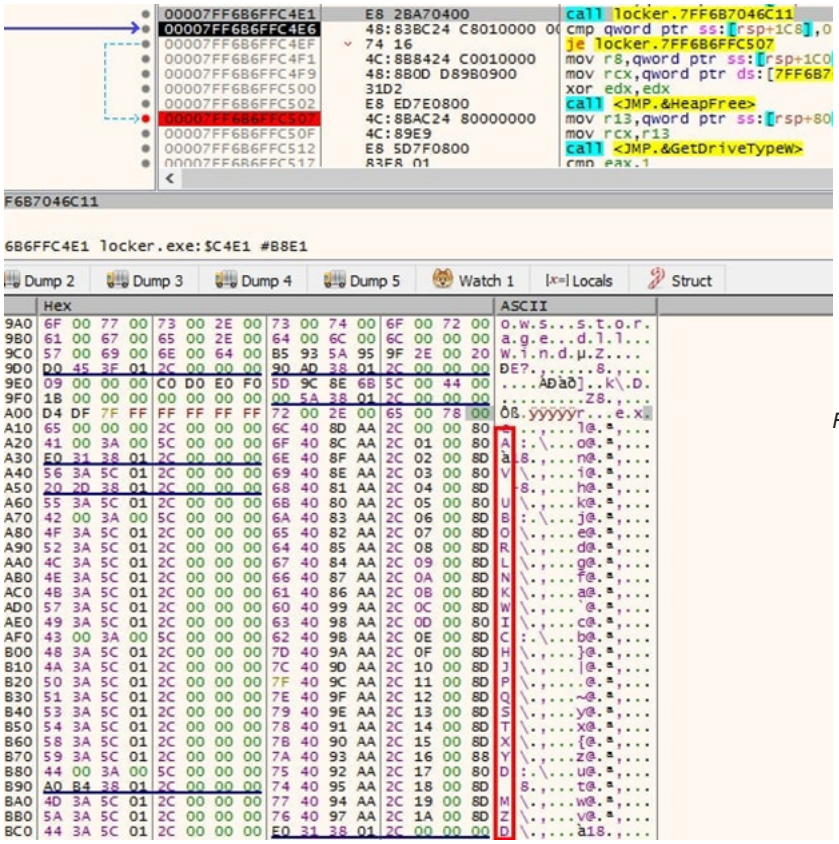


Figure 25: Finding of logical drives

Once the attached volumes are found, it calls **FindFirstVolumeW** and **SetVolumeMountPointW** to mount eventual unmounted volumes:

```
FirstVolumeW = FindFirstVolumeW(v422, 0x7D00u);
  if ( FirstVolumeW )
  {
    v429 = FirstVolumeW;
    hSCManagerc = v424;
    do
    {
      *(_DWORD *)cchReturnLength = 260;
      if ( !GetVolumePathNamesForVolumeNameW(v422, v425, 0x104u, (PDWORD)cchReturnLength) || !*(_DWORD *)cchReturnLength )
      {
        v430 = *(_QWORD *)&lpParameters[16];
        if ( *(_QWORD *)&lpParameters[16] )
        {
          --*(_QWORD *)&lpParameters[16];
          v431 = 3 * (v430 - 1);
          v432 = *(void **)(*(_QWORD *)lpParameters + 8 * v431);
          if ( v432 )
          {
            v433 = (__int64 *)(*(_QWORD *)lpParameters + 8 * v431 + 8);
            v434 = *v433;
            sub_13FED0120((__int64)Src, (__int64)v432, v433[1]);
            *(_QWORD *)lpMem = sub_13FEC8350(Src);
            *(_QWORD *)&lpMem[8] = *(_QWORD *)lpMem + v435;
            *(_WORD *)&lpMem[16] = 0;
            *(_DWORD *)&lpMem[24] = 1;
            sub_13FE96C11(lpRootPathName, lpMem);
            if ( *(_QWORD *)&Src[0].dwControlsAccepted )
              HeapFree(hHeap, 0, *(LPVOID *)&Src[0].dwServiceType);
            v436 = (void *)lpRootPathName[0].m128i_i64[0];
            SetVolumeMountPointW((LPCWSTR)lpRootPathName[0].m128i_i64[0], v422);
```

Figure 26: Mounting available volumes

After that, the operation of privilege escalation is performed though abusing the "*TrustedInstaller*" service to recover its access token. In this way, the malware is able to read and write files with the same privileges of **TrustedInstaller Group**.

```
00007FF6034ADAC0   31D2              xor edx,edx
00007FF6034ADAC2   E8 DD680800       call <JMP.&OpenProcess>
00007FF6034ADAC7   48:85C0           test rax,rax
00007FF6034ADACA   74 4B             je locker.7FF6034ADB17
00007FF6034ADACC   48:89C6           mov rsi,rax
00007FF6034ADACF   48:C78424 C0010000 0( mov qword ptr ss:[rsp+1C0],0   [rsp+1C0]:"TrustedInstaller.exe"
00007FF6034ADADB   48:89C1           mov rcx,rax
00007FF6034ADADE   BA 0E000200       mov edx,2000E
00007FF6034ADAE3   49:89F8           mov r8,rdi                         rdi:&"TrustedInstaller.exe"
00007FF6034ADAE6   E8 81670800       call <JMP.&OpenProcessToken>
00007FF6034ADAEB   85C0              test eax,eax
00007FF6034ADAED   74 20             je locker.7FF6034ADB0F
00007FF6034ADAEF   48:8B8C24 C0010000 mov rcx,qword ptr ss:[rsp+1C0]    [rsp+1C0]:"TrustedInstaller.exe"
00007FF6034ADAF7   E8 88670800       call <JMP.&ImpersonateLoggedOnUser>
00007FF6034ADAFC   89C3              mov ebx,eax
00007FF6034ADAFE   48:8B8C24 C0010000 mov rcx,qword ptr ss:[rsp+1C0]    [rsp+1C0]:"TrustedInstaller.exe"
00007FF6034ADB06   E8 116A0800       call <JMP.&CloseHandle>
00007FF6034ADB0B   85DB              test ebx,ebx
00007FF6034ADB0D   75 31             jne locker.7FF6034ADB40
00007FF6034ADB0F   48:89F1           mov rcx,rsi
00007FF6034ADB12   E8 056A0800       call <JMP.&CloseHandle>
00007FF6034ADB17   4C:89E9           mov rcx,r13
00007FF6034ADB1A   48:8D9424 00050000 lea rdx,qword ptr ss:[rsp+500]
00007FF6034ADB22   E8 6D680800       call <JMP.&Process32NextW>
00007FF6034ADB27   85C0              test eax,eax
```

Figure 27: Retrieving TrustedInstaller access token

Moreover, in the previous versions, we saw bat files and other escamotages for the erasing of the backup mechanisms provided by the Microsoft Environment.

In the fifth version analyzed, there are the following tricks:

vssadmin.exe delete shadows /all /quiet

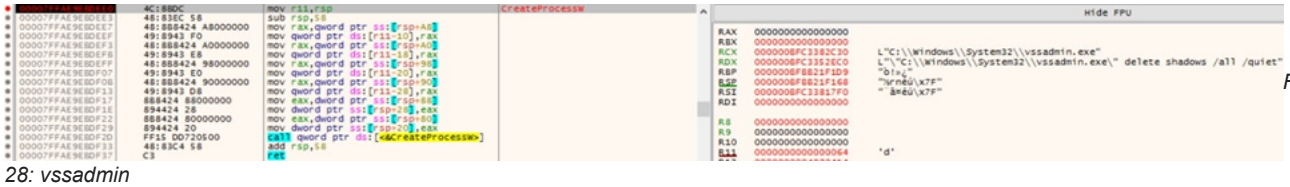


Figure 28: vssadmin

bcedit /set {default} bootstatuspolicy ignoreallfailures

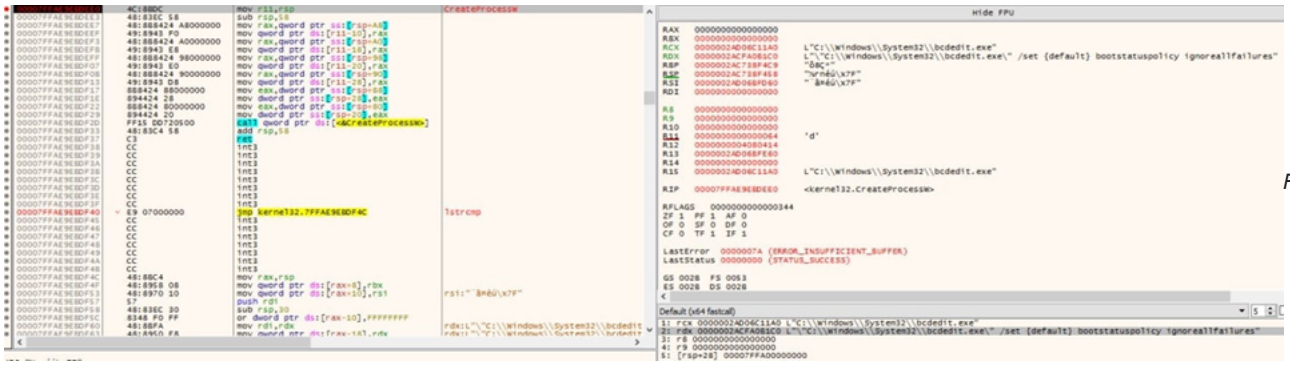


Figure 29: bcedit

wbadmin delete systemstatebackup –keepversions:3

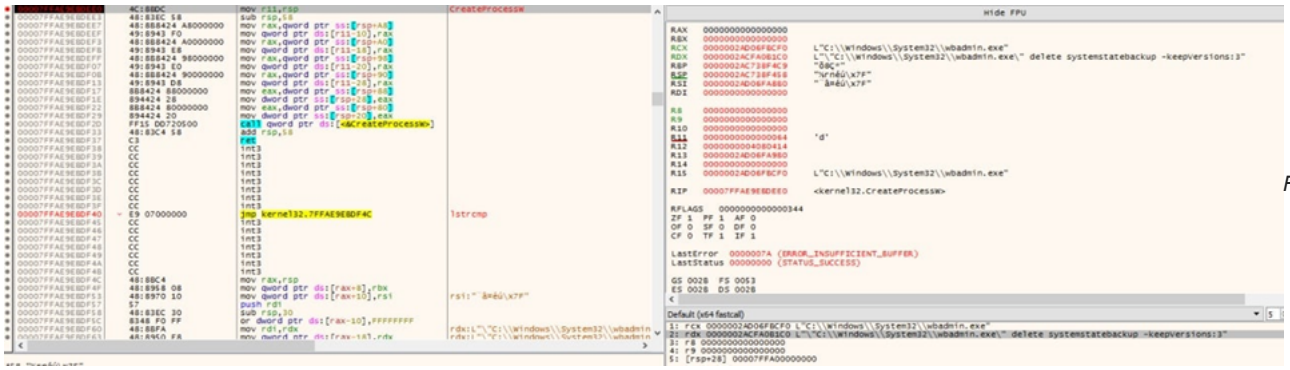


Figure 30: wbadmin

Conclusion

Hive threat actor is one of the most sophisticated active threats. It does not care about the target, the only objective is to maximize the illicit profits, even by causing the interruption of critical services. The continuous development of the ransomware payload should not be underestimated, and in the same way organizations must upgrade their cyber protections.

We at Yoroi ZLab believe that collaboration and sharing more information possible about attackers is the right way to pursue to defend these entities. We know that having to deal with these threats is challenging, so we are pointing to create the best expertise needed to handle such incidents whether they happen.

In conclusion, we need to create a solid and reliable strategy to defend our customers. we encourage our customers to make assessments and awareness campaigns for their employees. The goal of the Defence Center of Yoroi is to guarantee the best protection in every phase of the attack, starting from the continuous monitoring arriving to the Incident Response engagements.

Appendix

Indicators of Compromise

Hive v1

- 88f7544a29a2ceb175a135d9fa221cbfd3e8c71f32dd6b09399717f85ea9afd1 (Sample)
- d158f9d53e7c37eadd3b5cc1b82d095f61484e47eda2c36d9d35f31c0b4d3ff8 (shadow.bat)

Hive v2:

25bfec0c3c81ab55cf85a57367c14cc6803a03e2e9b4afd72e7bbca9420fe7c5

Hive v3

8a461e66ae8a53ffe98d1e2e1dc52d015c11d67bd9ed09eb4be2124efd73ccd5

Hive v3 Linux/FreeBSD

- 12389b8af28307fd09fe080fd89802b4e616ed4c961f464f95fdb4b3f0aaf185 (Linux)
- Bdf3d5f4f1b7c90dfc526340e917da9e188f04238e772049b2a97b4f88f711e3 (FreeBSD)

Hive v3 ESXI

822d89e7917d41a90f5f65bee75cad31fe13995e43f47ea9ea536862884efc25

Hive v4

33aceb3dc0681a56226d4cfce32eee7a431e66f5c746a4d6dc7506a72b317277

Hive v5.2

b6b1ea26464c92c3d25956815c301caf6fa0da9723a2ef847e2bb9cd11563d8b

Yara Rules

```
rule hive_v1_32_win
{
  strings:
    $1 = {648b0d140000008b89000000003b61080f86e401000083ec40e8?2f?feff8b04248b4c240485c90f8556010000b94100000031d231db8d2d?4??
6300eb0341d1e883f95a0f8f29010000a90100000074ed895c2434896c243c894c2424895424308944242884d44242c}
  condition:
    $1 and uint16(0) == 0x5A4D
}

rule hive_v1_64_win
{
  strings:
    $1 = { 65 4? 8b 0c ?5 28 00 00 00 4? 8b 89 00 00 00 00 4? 3b 61 10 0f 86 ?? ?? ?? ?? 4? 83 ec 40 4? 89 6c ?4 38 4? 8d 6c ?4 38
4? 8b 44 ?4 48 4? 89 04 ?4 e8 ?? ?? ?? ?? 4? 8b 44 ?4 08 4? 8b 4c ?4 10 4? 83 7c ?4 08 00 0f 85 ?? ?? ?? ?? 4? 89 44 ?4 20 4? 89 4c
?4 18 4? 8b 44 ?4 48 4? 89 04 ?4 90 e8 ?? ?? ?? ?? 4? 8b 44 ?4 48 4? 89 04 ?4 e8 ?? ?? ?? ?? 4? 8b 44 ?4 48 4? 89 04 ?4 0f 1f 40 00
e8 ?? ?? ?? ?? 4? 8b 44 ?4 48 4? 89 04 ?4 e8 ?? ?? ?? ?? 4? 8b 44 ?4 48 4? 89 04 ?4 0f 1f 40 00 e8 ?? ?? ?? ?? 4? 8b 44 ?4 48 4? 89
04 ?4 e8 ?? ?? ?? ?? 4? 8b 44 ?4 48 4? 89 04 ?4 0f 1f 40 00 e8 ?? ?? ?? ?? 4? 8b 44 ?4 48 4? 89 04 ?4 e8 ?? ?? ?? ?? 4? 8b 44 ?4 48
4? 89 04 ?4 0f 1f 40 00 e8 ?? ?? ?? ?? 4? 8b 44 ?4 48 80 78 48 00 74 ?? 90 0f 57 c0 0f 11 44 ?4 28 4? 8d 05 ?? ?? ?? ?? 4? 89 ?? ?4
28 4? 8d 05 ?? ?? ?? ?? 4? 89 ?? ?4 30 4? 8d 44 ?4 28 4? 89 04 ?4 4? c7 44 ?4 08 01 00 00 00 4? c7 44 ?4 10 01 00 00 00 e8 ?? ?? ??
?? 4? 8b 44 ?4 20 4? 89 44 ?4 50 4? 8b 44 ?4 18 4? 89 44 ?4 58 4? 8b 6c ?4 38 4? 83 c4 40 c3 4? 89 04 ?4 e8 ?? ?? ?? ?? eb ?? 4? 89
44 ?4 50 4? 89 4c ?4 58 4? 8b 6c ?4 38 4? 83 c4 40 c3  }

  condition:
    $1 and uint16(0) == 0x5A4D
}

rule hive_v2_v3_32_win
{
  strings:
//prenotify routine
    $1 = { 64 8b 0d 14 00 00 00 8b 89 00 00 00 00 3b 61 08 0f 86 ?? ?? ?? ?? 83 ec ?? c7 44 ?4 04 ?? ?? ?? ?? c7 04 ?4 ?? ?? ?? ??
e8 ?? ?? ?? ?? e8 ?? ?? ?? ?? 8b 04 ?4 8b 4c ?4 04 c7 44 ?4 4c 00 00 00 00 c7 44 ?4 50 00 00 00 00 89 04 ?4 89 4c ?4 04 e8 ?? ?? ??
?? 8b 44 ?4 08 8d 0d ?? ?? ?? ?? 89 4c ?4 4c 89 44 ?4 50 8d 44 ?4 4c 89 04 ?4 c7 44 ?4 04 01 00 00 00 c7 44 ?4 08 01 00 00 00 e8 ??
?? ?? ?? 8b 44 ?4 68 89 04 ?4 e8 ?? ?? ?? ?? 8b 44 ?4 04 89 44 ?4 48 8b 4c ?4 08 89 4c ?4 38 31 d2 eb ??  }
  condition:
    $1 and uint16(0) == 0x5A4D
}

rule hive_v2_64_win
{
  strings:
      $1 = {654?8b0c?5280000004?8b890000000004?8d44????4?3b41100f86????????4?81ec????????4?89ac??????????4?8dac??????????4?
b8??????????????????4?8904?4e8????????e8?????????4?8b04?44?8b4c?4080f57c00f1184??????????4?8904?44?894c?408e8????????4?8b44?4104?
8d0d????????4?898c??????????4?8984??????????4?8d84??????????4?8904?44?c744?408010000004?c744?41001000000e8????????4?8b84??????????
4?8904?40f1f440000e8????????4?8b44?4084?8b4c?4104?85c97e??4?89?c}
  condition:
    $1 and uint16(0) == 0x5A4D
}

rule hive_v3_v4_64_win
{
  strings:
    $1 = {4? 3b 66 10 0f 86 ?? ?? ?? ?? 4? 83 ec 30 4? 89 6c ?4 28 4? 8d 6c ?4 28 4? 89 44 ?4 20 0f 1f 00 e8 ?? ?? ?? ?? 4? 85 c0
0f 85 ?? ?? ?? ?? 4? 8b 44 ?4 20 e8 ?? ?? ?? ?? 4? 85 c0 74 ?? 4? 8b 6c ?4 28 4? 83 c4 30 c3 4? 89 44 ?4 10 4? 89 5c ?4 18 4? 8b 44
?4 20 e8 ?? ?? ?? ?? 4? 8b 44 ?4 20 e8 ?? ?? ?? ?? 4? 8b 44 ?4 20 e8 ?? ?? ?? ?? 4? 89 c3 4? 8b 44 ?4 20 e8 ?? ?? ?? ?? 4? 8b 44 ?4
20 e8 ?? ?? ?? ?? 4? 8b 44 ?4 20 e8 ?? ?? ?? ?? 4? 8b 44 ?4 20 e8 ?? ?? ?? ?? 4? 8b 44 ?4 20 90 e8 ?? ?? ?? ?? 4? 8b 44 ?4 10 4? 8b
5c ?4 18 4? 8b 6c ?4 28 4? 83 c4 30 c3 4? 8b 6c ?4 28 4? 83 c4 30 c3}
  condition:
    $1 and uint16(0) == 0x5A4D
}

rule hive_v5_32_win
{
  strings:
    $1 = {5589e553575681ec440400008b75108b7d0c89d3894dc88d85b0fbffff68000400006a0050e8????????
83c40c0fbec3b9abaaaaaa8b0485c0b949008945e889f0f7e1d1ea8d045229c683f60389f0f7e131c9d1ea8d04528d570229c68b45148955cc8975e48b00}
  condition:
    $1 and uint16(0) == 0x5A4D
}


rule hive_v5_64_win
{
  strings:
    $1 =
{41574156415541545565755534881ec880400004c89cd448844243789d648894c2450488b9c24f0040000488bbc24f8040000488d8c248800000041b80004000031c
00480fbec6488d0d??????
004c8b24c148b9abaaaaaaaaaaaaaaa4889d848f7e148d1ea488d04524989de4929c64983f6034c89f048f7e148d1ea488d04524929c6488b07}
  condition:
```

```
    $1 and uint16(0) == 0x5A4D
}

rule hive_v3_esxi
{
  strings:
    $s1 = "+ prenotify %s"
    $s2 = "Stopping VMs"
    $s3 = "(.+)\\.(.+?)\\.%s$"
    $s4 = "\\.(vm|vs)\\w+$"

    $c = {f3 0f 1e fa 55 4? 89 e5 4? 83 ec 20 4? 89 7? ?? 4? 8b 4? ?? 4? 89 c7 e8 ?? ?? ?? ?? 89 4? ?? 83 7? ?? 00 74 ?? 8b 4? ??
eb ?? 4? 8b 4? ?? 4? 89 c7 e8 ?? ?? ?? ?? 89 4? ?? 83 7? ?? 00 74 ?? 4? 8d 3d ?? ?? ?? ?? e8 ?? ?? ?? ?? 8b 4? ?? eb ?? 4? 8b 4? ??
4? 89 c7 e8 ?? ?? ?? ?? 4? 8b 4? ?? 4? 89 c7 e8 ?? ?? ?? ?? 4? 8b 4? ?? 4? 89 c7 e8 ?? ?? ?? ?? 4? 8b 4? ?? 4? 89 c7 e8 ?? ?? ?? ??
4? 8b 4? ?? 4? 89 c7 e8 ?? ?? ?? ?? b8 00 00 00 00 c9 c3}

  condition:
    (all of ($s*) or $c) and uint32(0) == 0x464C457F
}
```

This blog post was authored by Luigi Martire, Carmelo Ragusa of Yoroi Malware ZLAB

Download The Report