Hellhounds: Operation Lahat. Part 2

Positive Technologies : : 5/22/2024

Introduction

In November 2023, the team at the Positive Technologies Expert Security Center (PT ESC) released their first research report on attacks by the hitherto-unknown group Hellhounds on Russian companies' infrastructure: Operation Lahat. The report focused on the group's attacks on Linux hosts that relied on a new backdoor known as Decoy Dog. Hellhounds carried on attacks on organizations located in Russia, scoring at least 48 confirmed victims by Q2 2024.

As the PT ESC CSIRT team responded to an incident at a transportation company, they detected previously unreported attacks on Windows-based infrastructure, besides already-known TTPs (Tactics, Techniques, and Procedures) and attacks on Linux hosts. The new investigation also found that Hellhounds had been successfully hitting Russian companies since at least 2021. It is a known fact that development of the malware began at least as early as 2019.

The Hellhounds group compromises organizations they select and gain a foothold on their networks, remaining undetected for years. In doing so, the group leverages primary compromise vectors, from vulnerable web services to trusted relationships. The malicious actor presumably penetrated the infrastructures by using supply chain attacks.

It would often disguise its tools as legitimate software processes including Positive Technologies products.

The report describes previously unknown parts of the group's toolkit, their obfuscation methods, and lists indicators of compromise and malware sample detection signatures.

An extended version of the research report was first presented at the international information security cyberfestival Positive Hack Days 2.

First Stage (Decoy Dog Loader for Windows)

After successfully compromising a Linux infrastructure, an event we described in detail last year, the malicious actor made a successful attempt to compromise mission-critical hosts running Windows. Having gained access to the system, the attackers installed a service named "Microsoft Account Service" or "Microsoft Viewer Service", which ran the PE executable AccSrvX64__STABLE__2016-11-10.exe or R_TARIF.VIEWS_X86.EXE. Below is an example of the services.


```
{
"Name": "Microsoft Account Service",
"Caption": "Microsoft Account Service",
"Description": "",
"DisplayName": "Microsoft Account Service",
"PathName": "C:\\[REDACTED]\\accounts64\\AccSrvX64__STABLE__2016-11-10.exe",
"ProcessId": 5092,
"Started": true,
"State": "Running",
"SystemName": "[REDACTED]",
"TimeLine": "2024-01-02T21:14:53.132165Z",
"ModuleName": "Win32_Service"
}
```

```
{
"Name": "Microsoft Viewer Service",
"Caption": "Microsoft Viewer Service",
"Description": "",
"DisplayName": "Microsoft Viewer Service",
"PathName": "C:\\[REDACTED] \\R_TARIF.VIEWS_X86.EXE",
"ProcessId": 5548,
"Started": true,
"State": "Running",
```

"SystemName": "[REDACTED]",
"TimeLine": "2024-01-03T22:04:30.5586058Z",
"ModuleName": "Win32_Service"
}

Interestingly, the malicious actor's activity in the compromised organization's Windows-based infrastructure began amid the New Year's holiday season on January 2 and 3.

The executable file size is 17 KB. After the service is started successfully, the sample decrypts a list of domains inside the .rdata section and then attempts to resolve the resulting domain names.

Each encrypted domain begins with an FF byte. Encryption uses a simple algorithm based on two operations: xor and subtract. Decryption involves the number of the character in the row and the row number; row character numbers start at zero.

```
for ( i = 0i64; i != decrypted_domain_length; ++i )
  decrypted_domain[i] = encrypted_domain[i + 1] ^ i ^ (domain_number - 15);
```

Figure 1. Encryption algorithm

The domains have the following format:

```
[-][!][!][...]<domain>
```

The "-" option means the domain does not have to be resolved. If it could not be resolved, the loader moves on to the next domain on the list. The "!" option is only used together with the "-" to show the number of resolve attempts that were made before the domain was skipped. The number of resolve attempts is calculated as 2^n, where n is the number of consecutive "!" options. If the option is missing, only one resolve attempt is made.

Domains in the configuration are used when obtaining a part of the key for payload decryption. They also can be used for generating legitimate-looking traffic and getting around sandboxes.

A superficial dynamic analysis may suggest that domains used at this stage are C2 servers. However, a detailed analysis shows that both domains and subsequently obtained IP addresses are used for key generation only and possibly, for disguising as legitimate utilities. Besides, the malware is notable for its ability to use non-existent subdomains located in valid domains, such as mp0.ptsecurity.com. While this may create a semblance of legitimacy, the domain is certain not to be resolved.

A domain with a "-" option is used for generating traffic but not a key. A domain like that must not be resolved, or alternatively, it is resolved after the right domain. One of the domains must be resolved and have a static IP address—this is what will be used for generating a key. The malicious actor notably used this feature as a kill switch to shut down the malware in a target system.

After all domains in the configuration are decrypted and resolved, the loader proceeds to decrypting the next block.

Figure 2. Block in the .rdata section

The block has a fixed size of 256 bytes, and it is encrypted with the CLEFIA algorithm in CBC mode. It contains the path to the main backdoor. The key is generated as follows: the name of the executable file minus the final zero is uppercased, and the byte-coded IP address is appended to it. The resulting byte string is hashed with SHA-3 to produce a 256 byte output. The first 16 bytes are used as the key, and bytes 5 through 20, as the initialization vector. Example of key generation.

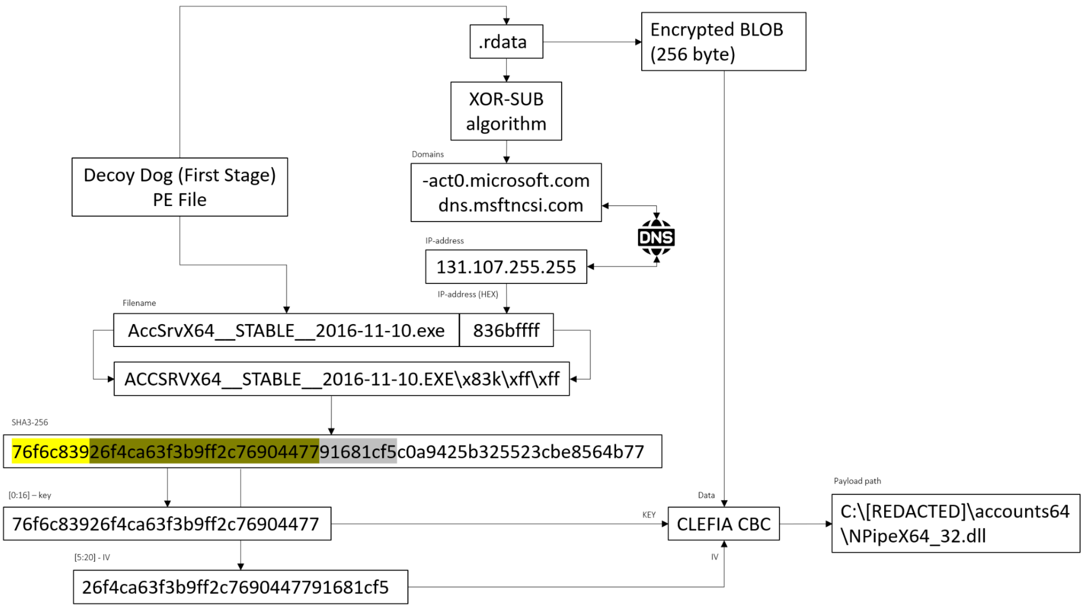

Figure 3. Second Stage (Decoy Dog for Windows) path decryption algorithm

After decrypting the path to the backdoor, the loader reads and decrypts it in the same manner, by using the same key and initialization vector, and then passes control to its entry point. Interestingly, unlike the Linux sample, the Windows malware does not check the integrity of decrypted data.

The backdoor has the MZ signature replaced with HE, and the PE signature, overwritten as a random 4-byte sequence.

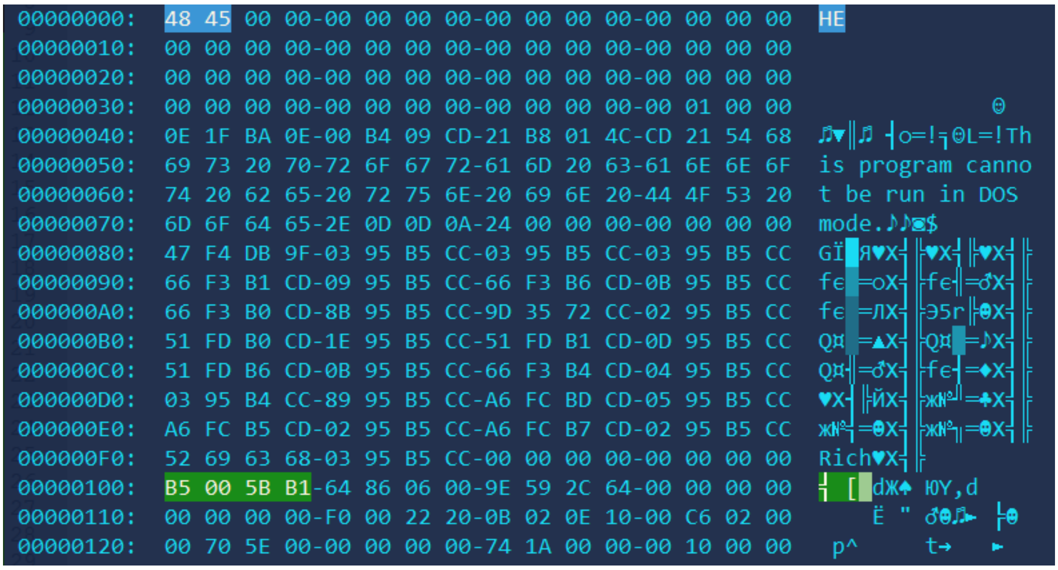

Figure 4. Fragment of Decoy Dog

The malicious actor invested a lot of effort in disguising its activity on the hosts that it compromised. To do this, they imitated MaxPatrol SIEM and Microsoft services.

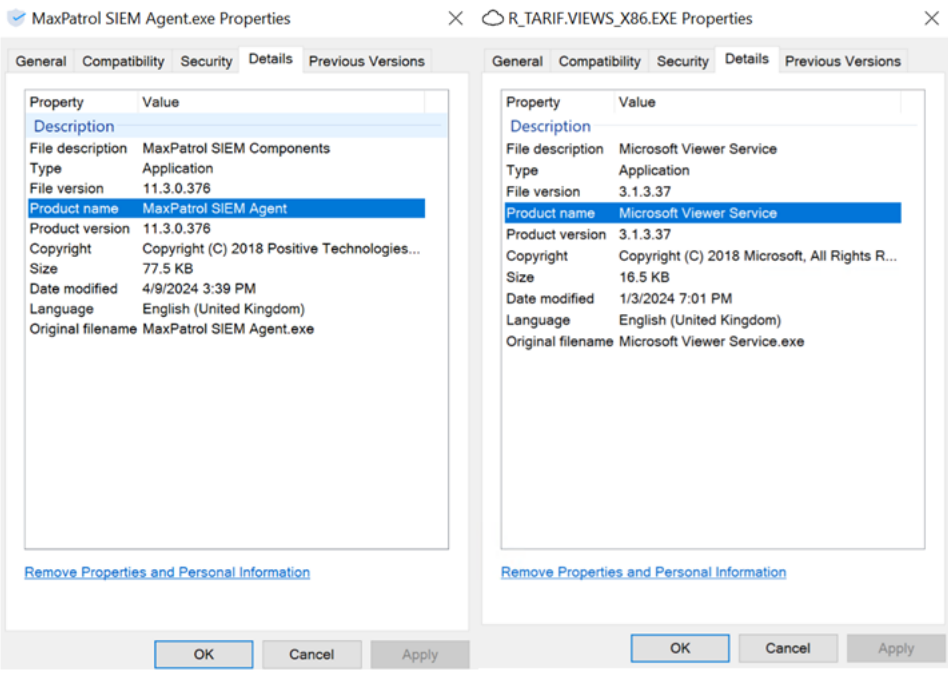

Figure 5. Information about the Decoy Dog Loader files for Windows

Tellingly, the Linux samples were virtually unusable unless they passed a machine-id check, that is, the malware could not be run without a valid identifier. The Windows samples do not contain a check like that, although they do check the executable name, which never matches the original filename in the metadata, a weaker check. If the IP address changes, the researchers can use PDNS (Passive DNS) services.

Second Stage (Decoy Dog for Windows)

The decrypted payload is all but identical to the Decoy Dog version for Linux examined earlier. The backdoor is based on the open-source project Pupy RAT.

All of the samples we managed to discover used the C2 server net-sensors[.]net and the DGA domain dynamic-dns[.]net. Neither of the samples had a dynamic configuration.

Configuration example:

```
{'debug': False, 'launcher': 'dnscnc', 'launcher_args': ['--domain', 'net-
sensors.net', '-E', 'dynamic-dns.net'], 'delays': [(10, 5, 10), (50, 30, 50), (-1,
150, 300)] [REDACTED] 'cid': 61336226}
```

Below is a detailed chart showing how Decoy Dog works on Windows hosts.

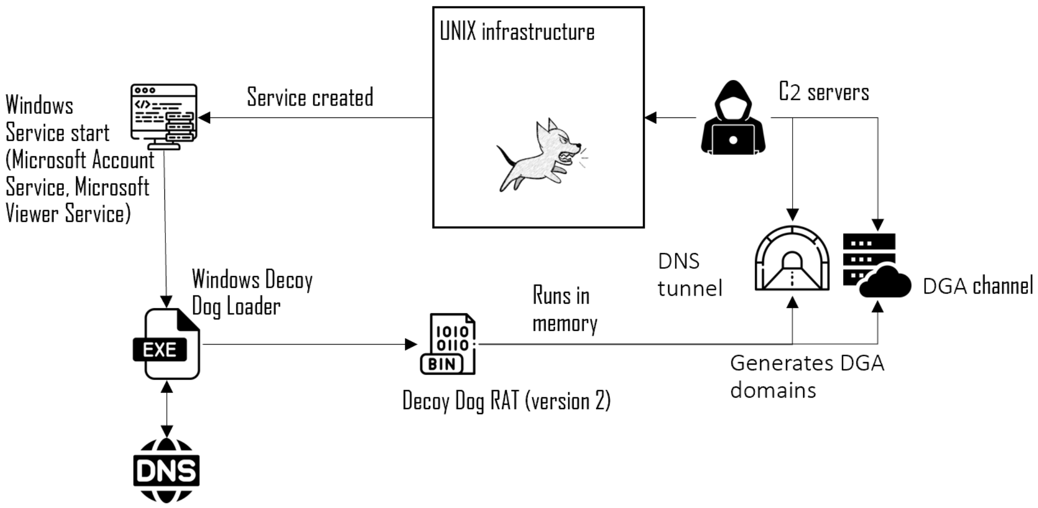

Figure 6. Detailed chart showing how Decoy Dog for Windows works

In the course of our research, we detected samples targeting Windows, the oldest of these compiled on 29.11.2019, and the newest one, on January 3, 2024. Besides Decoy Dog, the malicious actor made use of the well-known Sliver framework with the C2 server 31.184.204[.]42 (ns2.maxpatrol[.]net). Similar samples are examined in detail in "Sliver Implants under a Lens: Extracting the Configuration and Other Useful Data". The table below shows all of the Windows samples we obtained.

Date	Description	SHA-256	Name
29.11.2019	First Stage (Decoy Dog Loader for Windows): test version	9a977571296ae1548c32df94be75eec2a414798bee7064b0bf44859e886a0cfa	testvec.exe
14.07.2022	First Stage (Decoy Dog Loader for Windows)	4d30fd05c3bdac792e0a011892e2cad02818436484e81b6de6a02928149bc92d	MaxPatrol SIEM Agent.ex
30.11.2022	First Stage (Decoy Dog Loader for Windows)	e27d1bab901c1bb414d0849c5c132faa8c7c6a61357d9627a7d2785270034793	Microsoft.exe
29.01.2023	First Stage (Decoy Dog Loader for Windows)	31b21de71f2162e8da1be8483f3a5d019b0c817832bc11a9f307b6b36821ca54	–
16.04.2023	First Stage (Decoy Dog Loader for Windows)	18d4a3a92b24b2ad75115a44fe2727081316eca346499a4aa00aa13713cf00cb	–
06.05.2023	First Stage (Decoy Dog Loader for Windows)	9a96c7b0595f628027c4f4caeece475ef742c420adf2fde8df934c6ce6481fb5	–
16.08.2023	First Stage (Decoy Dog Loader for Windows)	d9a8151aff9d1c061826a9812ed9a6600805c74a519df333513fd4a79d2d4e61	NtpService.exe
06.11.2023	First Stage (Decoy Dog Loader for Windows)	07fe71b256c1c913b0f3e3fa67e53d21a3d1f499beb4e550597f5743797a77c4	Apache ActiveMQ.exe
08.11.2023	Second Stage (Decoy Dog for Windows)	e19dc185e99cfdc0c25f18fb34ffabff2a4877d6d5843e4c67c05ce182f9780e	NPipeX64_32.dll

Date	Description	SHA-256	Name
08.11.2023	Second Stage (Decoy Dog for Windows)	106436a4fafe00112b19b1374456c1746b988950b71d700680088d74494e4936	r_tarif.dll2Qur
27.12.2023	Sliver	510da6d88ae4dd51d62796023a18b39db08a016ee4ee7178b1afdc91c58f9e1e	–
27.12.2023	Sliver	6cb2979aa1fddd42df2ba596f705ce9bbdb2ec246649218d598d779769857c21	–
02.01.2024	First Stage (Decoy Dog Loader for Windows)	1b8b4be020d3350d025c7a245eb0d7166ff2c329dc92af175ef0499cba583071	AccSrvX64__STABLE__2 11-10.exe
03.01.2024	First Stage (Decoy Dog Loader for Windows)	a03e2ca143e867a99e2bc73bd4e5c2dd078a9f671aa0a4ce9611a8bc39a769e2	R_TARIF.VIEWS_X86.EX

Most of the samples contain the domain dns.msftncsi.com in their configurations, a test server for the Windows Network Connectivity Status Indicator, NCSI.

After examining the configurations of all samples we obtained during the research, we identified SSL certificates that the backdoor used to encrypt its connections with remote hosts. The certificates contained the earliest notBefore option at the end of 2021 (12/26/2021 at 21:51:52), and the latest option, on 11/8/2023 at 13:48:36. This places the campaign start at the end of 2021. Certificates were issued for one and three years from the time the images were generated. This certificate generation algorithm is implemented in the public Pupy RAT project.

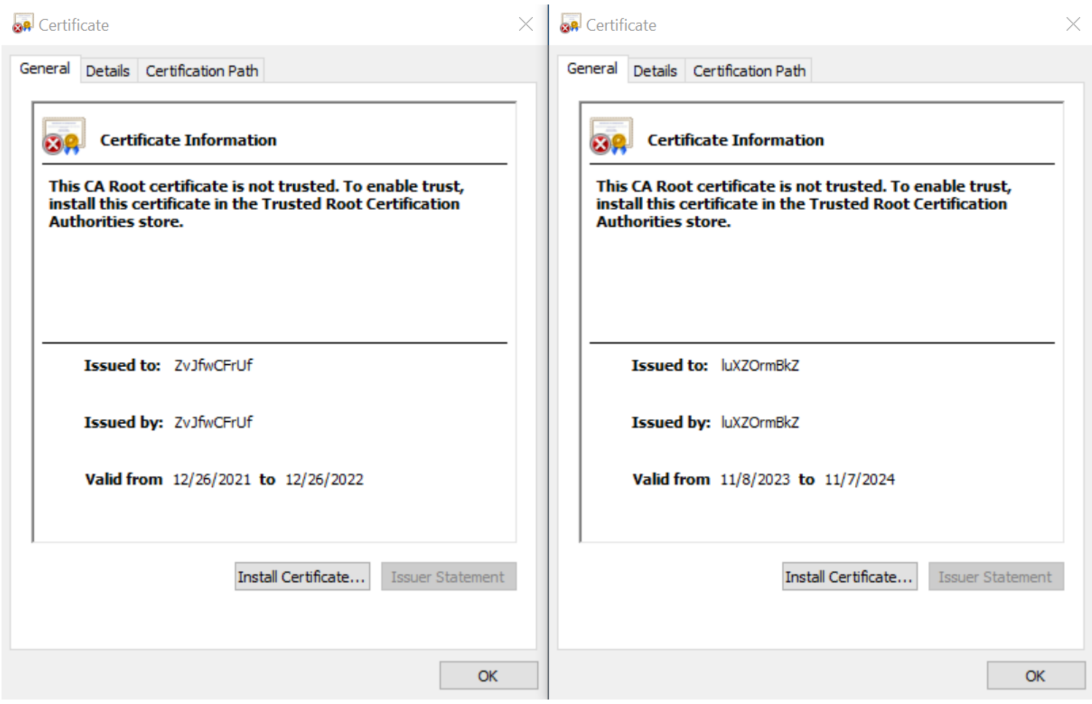

Figure 7. Examples of certificates

After analyzing all of the samples we found, we compared their features, the issue dates of the certificates in the configuration, and the VirusTotal upload dates. The relevant feature set appeared in between these dates—this time range is marked dull blue in the image. This data can be used to tentatively distinguish two versions of Decoy Dog. Compared with Pupy RAT, the project migrated to Python 3.8, added new transports, and received a DGA mechanism. The second version, created between April 2022 and February 2023, gets a telemetry scriptlet described in detail in the previous article, a dynamic configuration, and a Special launcher to run as a server on the local machine.

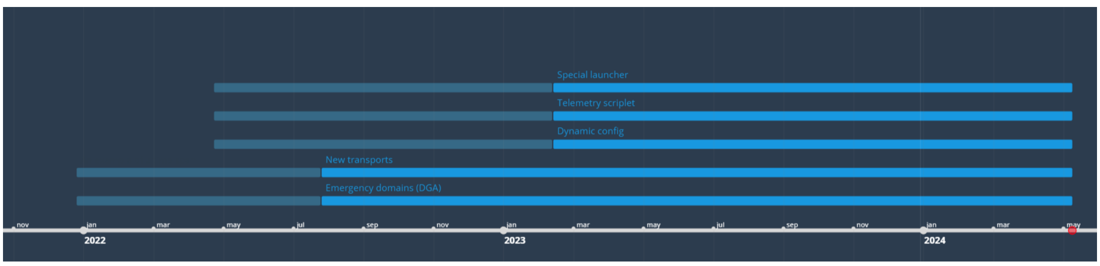

Figure 8. Timeline of new features

The earliest Decoy Dog loader sample, compiled at the end of 2019 (11/29/2019), deserves special attention. The sample is the original version of the loader whose code contains several debugging strings. This suggests that the development of the Decoy Dog loader began in 2019.

```
if ( SHGetFolderPathA(0i64, CSIDL_PROFILE, 0i64, 0, FileName) < 0 )
  ExitProcess(0x7Eu);
LODWORD(v0) = 0;
for ( i = FileName; *i; ++i )
  v0 = &i[1i64 - (_QWORD)FileName];
v2 = 'A';
log_path = "\\AppData\\\\Local\\\\Temp\\\\loader.log";
v4 = '\\';
v5 = &FileName[(unsigned int)v0];
```

Figure 9. Generating a log path

```
*(_DWORD *)&payload_path[(_QWORD)v19 + 44] = 'NIB.';
write_to_log("Input file: ");
write_to_log(Filename);
write_to_log("\n");
write_to_log("Basename: ");
write_to_log(v13);
write_to_log("\n");
FileA = CreateFileA(Filename, 0x80000000, 0, 0i64, 3u, 0x80u, 0i64);
FileSize = GetFileSize(FileA, 0i64);
lpAddress = VirtualAlloc(0i64, FileSize, 0x3000u, 4u);
```

Figure 10. Downloading a payload

3snake

The malicious actor used a modified open-source 3snake utility to obtain credentials on hosts running Linux. To reduce excess functionality and evade signature detection, the command-line start option was disabled in the utility, which left just demon mode. Additionally, the utility ignores "-o" values, instead using the hardcoded path /var/log/apt/term.log.gz for outputting compromised credentials.

Unlike the original utility, the path to the file in the sample and intercepted data are encrypted with the RC4 algorithm. The utility can intercept SIGINT, SIGQUIT, SIGHUP, SIGPIPE, SIGTERM, SIGSEGV, SIGBUS, SIGILL, and SIGCHLD system-call interrupts. It also adds intercept_openldap to the already-available intercept_ssh, intercept_sudo, intercept_su, intercept_ssh_client, and intercept_passwd functions. This is how the malicious actor stole a number of credentials for further movement across the network.

Function name	Segment	Start
intercept_sudo	.text	0000000000002B80
intercept_su	.text	00000000000030D0
intercept_ssh	.text	0000000000003580
intercept_ssh_client	.text	0000000000003810
intercept_passwd	.text	0000000000004FE0
intercept_openldap	.text	0000000000005450

Figure 11. Intercept functions in 3snake

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int v4; // eax
  char *v5; // r14
  size_t v6; // rax

  signal(2, exitsig);
  signal(3, exitsig);
  signal(1, exitsig);
  signal(13, exitsig);
  signal(15, exitsig);
  signal(11, exitsig);
  signal(7, exitsig);
  signal(4, exitsig);
  signal(17, handlechild);
  if ( geteuid() )
    needroot();
  while ( 1 )
  {
    do
    {
      v4 = getopt(argc, (char *const *)argv, "do:");
      if ( v4 == -1 )
        daemonize((unsigned int)argc, argv, envp);
    }
    while ( v4 == 100 );
    if ( v4 != 111 )
      break;
    v5 = optarg;
    v6 = strlen(optarg);
    outfile = (char *)calloc(v6 + 1, 1uLL);
    __isoc99_sscanf(v5, "%s", outfile);
  }
  return 0;
}
```

Figure 12. Main function in the modified 3snake utility

Initial Access

In two incidents, the attackers managed to penetrate the victims' infrastructure via a contractor. By compromising SSH login credentials, the malicious actor got in and installed the Decoy Dog backdoor.

We also managed to obtain content from the C2 server net-sensor[.]net and discovered that the malicious actor disguised Decoy Dog as ISO images for the iMind online meeting, video conferencing, and webinar service. Unfortunately, we could not find out under what pretext and how exactly the malicious actor made the victims run one of the ISOs. Note that in September 2023, the National Computer Incident Response and Coordination Center issued a notice about an increased frequency of computer incidents associated with exploiting a vulnerability in the iMind video conferencing service and recommended updating iMind to version 3.19.

Figure 13. C2 folder listing

mind-live_3.12.31+23.07.23.01.30.iso	26.12.2023 13:36	Файл образа диска	4 192 КБ
mind-live_3.12.30+23.06.23.01.95.iso	26.12.2023 13:36	Файл образа диска	4 192 КБ
mind-live_3.12.30+23.06.23.01.90.iso	26.12.2023 13:36	Файл образа диска	4 192 КБ
mind-live_3.12.30+23.06.23.01.80.iso	26.12.2023 13:36	Файл образа диска	4 192 КБ
mind-live_3.12.30+23.06.23.01.70.iso	26.12.2023 13:36	Файл образа диска	4 192 КБ
mind-live_3.12.30+23.06.23.01.63.iso	26.12.2023 13:36	Файл образа диска	4 192 КБ
mind-live_3.12.30+23.06.23.01.62.iso	26.12.2023 13:36	Файл образа диска	4 192 КБ
mind-live_3.12.30+23.06.23.01.61.iso	26.12.2023 12:38	Файл образа диска	4 192 КБ
mind-live_3.12.30+23.06.23.01.50.iso	26.12.2023 12:38	Файл образа диска	4 192 КБ
mind-live_3.12.30+23.06.23.01.40.iso	26.12.2023 12:38	Файл образа диска	4 450 КБ
mind-live_3.12.30+23.06.23.01.30.iso	26.12.2023 12:36	Файл образа диска	4 450 КБ

Figure 14. Contents of the imind folder, ISO images containing Decoy Dog

Victims

As a result of the research into the group's activities, we detected a number of previously unknown attacks on organizations located in Russia: the number of confirmed victims more than doubled, reaching 48. At the time of preparing part one of the research report, we were aware of 20 Hellhounds victims. An analysis of the new attacks suggests that, in addition to focusing on the public sector, the attackers have been harassing Russian IT companies, most of these being contractors for critical organizations. These companies were presumably targeted for trusted relationship attacks. The up-to-date victim breakdown by industry looks as follows:

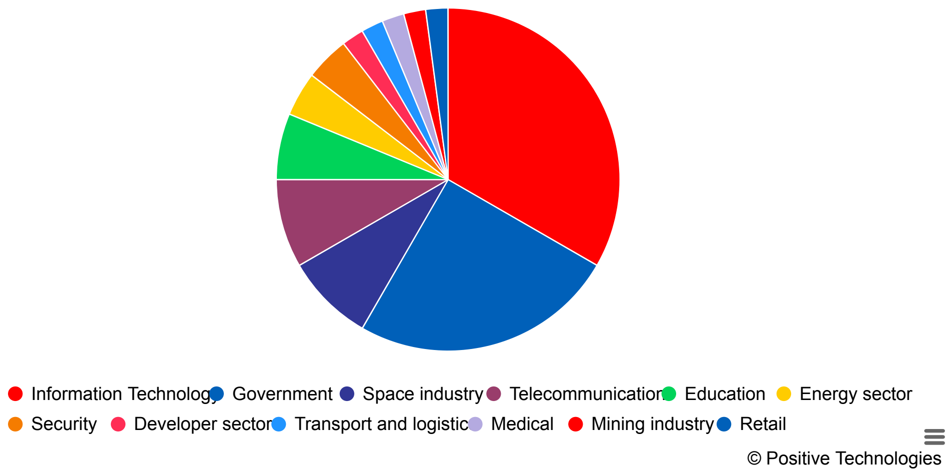

Figure 15. Victims by industry

Conclusion

The Hellhounds group has continued to attack Russian organizations into 2024. Our investigations show that the malicious actor uses a variety of techniques for compromising and gaining control over company infrastructures, and that it began developing its toolkit at least as early as 2019.

The attackers have long been able to maintain their presence inside critical organizations located in Russia. Although virtually all of the Hellhounds toolkit is based on open-source projects, the attackers have done a fairly good job modifying it to bypass malware defenses and ensure prolonged covert presence inside compromised organizations.

Authors: Aleksandr Grigorian and Stanislav Pyzhov at Positive Technologies

The authors would like to thank the Incident Response and Threat Intelligence teams at the PT Expert Security Center for their help in preparing this article.

The sections below contain information about all of the samples we obtained and the up-to-date TTPs.

Verdicts by our products

MaxPatrol SIEM

Modify_and_Start_Remote_Service
Service_Created_or_Modified

PT Sandbox

apt_linux_ZZ_DecoyDog__Trojan__FirstStage
apt_linux_ZZ_DecoyDog__Backdoor__Pupy
apt_linux_ZZ_DecoyDog__Backdoor__EncryptedPayload
apt_mem_ZZ_DecoyDog__Backdoor
apt_win_ZZ_DecoyDog__Trojan__FirstStage
tool_multi_ZZ_3snake__HackTool
tool_win_ZZ_Sliver__Backdoor
tool_multi_ZZ_Sliver__Backdoor

PT Network Attack Discovery

SUSPICIOUS [PTsecurity] Possible DecoyDog DNS Tunneling sid: 10010052
SUSPICIOUS [PTsecurity] Possible DecoyDog DNS Tunneling sid: 10010053
REMOTE [PTsecurity] Possible PupyRAT TLS JA3 fingerprint sid: 10011346
REMOTE [PTsecurity] Possible PupyRAT sid: 10008450-10008452
REMOTE [PTsecurity] PupyRAT HTTP Echo Scan sid: 10006391
REMOTE [PTsecurity] PupyRAT HTTP sid: 10006389
REMOTE [PTsecurity] PupyRAT TCP ping sid: 10006390
REMOTE [PTsecurity] PupyRAT SSL Cert 10011347-10011351, 10004069, 10008396-10008397, 10008449

Analysis tools

Our tools for Decoy Dog analysis are available on GitHub.

IOCs

File indicators

Name	MD5	SHA-1	
		First Stage (Decoy Dog Loader for Windows)	
testvec.exe	7e0c85852b2cd932626fcf284ca72978	c8ccf6e20cde537f3da64aebd1f80b144a4c8e0a	9a977
MaxPatrol SIEM Agent.exe	2c016c91181d4182a16845725bf0b315	2be016b6b0dd9d57f2985a6ad0df85f5538d9623	4d30fd
Microsoft.exe	4479cc492fa443af1461ebd768dcd1c3	5ebf1dbcd5e16bcd4695777a7931ff4dc13d586a	e27d1
–	ef6c7eb5518d58bc0b921d37265b0db4	c0fd9928b1755c047529a0b91517882bf74bc5e4	31b21
–	3dc4391eb6170c26336938839246022f	c4ef4c518c44eda803200b8f9d080c0f1ff3ed15	18d4a
–	321e4b64bcedc76a89cca86853d30c09	b1fceda9a56d17fd1520105a6d52fdf868c4cead	9a96c
NtpService.exe	9200c356b485ca61ec88258f0800657a	dc76c7586e1946ac120111d3a35937526a7cf140	d9a81
Apache ActiveMQ.exe	b8932033b53ca08967100c58e12126be	6f30131181d81129c2f59d050214f47a6eedabbe	07fe71
AccSrvX64__STABLE__2016-11-10.exe	8d6e4cd33145ae084aa184fd0875c8f6	fc5936e0e290f2f41a46eb14c05500a4236ac0c7	1b8b4
R_TARIF.VIEWS_X86.EXE	e908da6041ae249f478bb22ac05e4b18	83c8168f7706148a6f28145872a7f3bf01037239	a03e2
		Second Stage (Decoy Dog for Windows)	
NPipeX64_32.dll	10be9ca61ef86589951ddcfddc3d9672	b3a6f0d8daf0347f56e95bf56cb60a7ea6f711cb	e19dc
r_tarif.dll2Qur	914f932feb7e08f3e0396e40b8ea46e7	54984e656f2bf1ed874b8b281d5abacdd517e51b	10643
		Decoy Dog for Linux	
systemd-inputd	c89d431abb6b5cc28c86196fbf898684	06335756b2a9afcf4147af25b06e30f63e5d52b9	bca6d
–	2a9137f615fa56f9ae11fa7c17963dad	a1790420cb2f546a79ddaeefacfd3b3a3b781e7c	9d909
salt-slave	485ad3a834d81e63be6c03e94371c007	be428ac644a1cd1173f9c2a8b5db3c5fb38f795b	299a7
snapmount	5e672d6d5c2fc6190bd670409b987dfd	ee7ce10b16d4052cf15c897d98a9e286ab63c30b	75bf7d
plymouthd	04fcea4bf75070e47f5f3e7e6958995f	cb883f4ea73eef125d9ba2b945ed1797a679ca7e	8184a
crond	b28b70b981a3b8e98874d23b24fc7dbd	c3669cc6fcc8a4eed8c3cad540a8f5402e4ddb79	83a29
mysqlrestore	15ebf623c05744403a163bd958522511	b7724bfa0041c0ae9882d880669751e290f6e88e	e67c5
md5sum.pm	bf27f6608cea8343c287b355244762e9	e46c422e5336c499b852ca77b4ab97b2607e54bf	04241
dtmf	bc0200af1ac2e44cdeefcb9907f4d1d3	24e7b9e904e90bfc9a9aadd8e347512ccbc895f6	2c726
atd	adac1dc0ec3dcf28157ab09d35d0cfcc	dd053b9cb14429cd4eec1b36e1a87f0a47289193	07dfb5
–	5be93fc5c858c3474bbfbc2555843966	3ab8a4e40f91febfdc2e6d69e162e3efc8b8b448	5264d
UPDATE.SH	885fa41b7e8e7d033cd01ee2e224cca2	a864cf53550d6daf38149d345d4563b65dd8580a	e42e4
atd	9f29794effd56e4075bb9f6e28b14678	01e71387a3ab05d73caed5435a8437faa8b66198	025d9
atd	f19890d3f004cb9ae23398a006e358f5	ac0fe4a4a400265a7d6a68a558443dfe77e0dfb4	d53fe0
atd	b8b11cfde33f285402ac17c50e89ce5e	8107ca980e32c8c905aa86c81c20fe799181bef8	95172
dcrond	6f40bc303944be1e322dcf5c40e3cde8	286dc3e3a055e37f47ceef45cf0fa55a6ab10111	dd83e
dcrond	8816c53603205717e5f1269385841784	b79b0fba2e698ef78747fb412cfbab3364fe3125	f11afd
TNTb	f09c0d5883a221d2e5f762480e946a78	506386147d393cef81019dda55ac85125914c6be	0eb2c
–	5f721ea01a017832be0bc4ed60f73f9d	39a54217868490ce71d6d0eaf6b9b2a2d747b3ca	30617
ucs-25.1.1034-debian10.tar	6f18d4f75e0cb13dbb868ce7c6fe8ab8	b86ac0e9c1d0ef17c9f7ec406d51d4b2ed08ff67	f1aa7c
crond	639826f50120006342e23a409ff6fa70	b0827b53e4d2a3d53f3ab467157f17377a243eaf	30fd37
nmbtrapd	250af8e186b4d72b70036f090e9aee25	98c4d06e1c09907c3a4734668fef1ccd2a5ffcbb	5ab70
nmbd	67aee8a9d41240c462ee7d7023977d84	eaf5fc2f0d9a84ff77008f805be4025df1085c20	66b7c

Name	MD5	SHA-1	
dtmf	cd10a6c402c6ccc870afa0001409c27f	943918176f941b162d668ea9642ba63d51450ff9	cb1993
md5sum.pm	9fb96e93ba9962919b261ae7dfe2b120	8f943b9f82892292162eb7964da3c9168df28116	d89671
nmbtrapd	bc7b5aa2a7f1e178fa8997c8d76ef041	f4c1f2882e20792463638cb75c4bb64e7aaf0401	e38dcc
rpc_lan.so	093f35facc67cee3a8c2cca8be8b2d6a	2c568ee8524a72cee2ae3002039f846988bea670	f466ec
nmbd	b2538dbf30dc3acb95930394e0ff3498	50105c6e64ccf058d604cdcf9123ed8bf163c41b	fd7298
aptitude-common	0e22f3587c519c1f0e4fc57a04d66edb	d7f31bbb9a7cbe911f5ae3253e650b1fa7cc4b4c	b3538
UPDATE.SH	537b8e319ef65435740b3e0c28722925	226d3a29149e36690b50b93beabed4481b1a48e6	c67f28
–	4f2529e5be66a80e44acaeaa418b575c	15a6fdc79f0724d4c3b18742e5f1d73fc6839ac4	6da74
dcrond	e35199eacd0bbc06cfb2c72e14f7a659	67006e298844b578cce9888c243640f7e1f2e7c7	00625
dcrond	453833594493c5064eeff8210d571224	fb0f1226903dde243ef08c26ec0c5d7880e9c291	7f55c7
dcrond	f28c7c354b9e27e8908dd0b8dc7da01b	444a7b477e6f7bd0d9be7add79b3e0415566169c	d59fcb
atd	74ed22250182d13df4e1ad4b4f91d519	7c440e9421c26fe7b73ae8e213ef58b3b615ed6f	25ff8d
atd	6218ad1e81b1cd9364bbe0059b99bc9a	27fedf90846efe5357e11c53d87612fbf6c421d4	82746
atd	e1a93ced3a55b34a54b5ac0dd095da59	2ba1a6808db393296a08968c220b193fae42c21b	ee8dd
smartmond	d80b3ff086aed177bb87c317188b92c2	2b1b5b4c7f9d4a963b0ec92a5eb2e28cb6cdc0b7	ae6c7
dcrond	cc7a6656832b6929722b1c38cc14b550	4450d904f695dd51eaea24e4e449707707c7852e	b21e9
atd	b514157f9b8cbb08d476e838171050d8	e3ce85ababbc7b8b4291551abb9f0928caeb2646	121ab
dcrond	33f3dd60e87aafd96adf62fcb5af725f	685ecfba19ad58f81acbb62a3fc9010128bf2000	64af32
atd	5da97bbd438a030b0427a15c69af0037	8daf7589f2e417363e3cfbc714fc9b299f54a36e	834d7
dcrond	84f2fa4d139ac10124f915584dda6476	26199b999facada1d6dafe78524321e575621d73	33e90
htop	6ee38226efedabbf794a37d0c972702c	3ca21bde29ff0744edacd611d82b50d297bb447a	494c8
systemd-crgoupsd	26e10db16c4b00c9d4afa1d3f2c5f080	4811d92b307a2929b25b39638b35f3e5692f4451	49cda
		Sliver	
AzimuthF.exe_	b0b2176187e24710ad9b4fbbb38573b1	db3ea044e32773c12d67a49588f5a12aae09e257	510da
AzimuthF_2.exe_	fd13efb096377f8bbf8b754874c40262	480c2a12dfc1900ab9bad635caf6c507a300623c	6cb29
		3Snake	
db-healthmon	18417672efbe00f3ecdd700c442137fd	ac469df608ef049708ba6efe72f4493ac20cdfd0	1b7d2
		ISO Images	
mind-live_3.12.30+23.06.23.01.30.iso	8a1834e81ffb4ded5b818db7db8e543b	9206f83e69c53c4460a30cc4046e59f50e25a1ad	0d6d8
mind-live_3.12.30+23.06.23.01.40.iso	8a1834e81ffb4ded5b818db7db8e543b	9206f83e69c53c4460a30cc4046e59f50e25a1ad	0d6d8
mind-live_3.12.30+23.06.23.01.50.iso	6703e425619a766ab521109885b51248	1d3ab04ace6895b042beb2d7ccfcd6c6cf5e620c	c6207
mind-live_3.12.30+23.06.23.01.61.iso	6703e425619a766ab521109885b51248	1d3ab04ace6895b042beb2d7ccfcd6c6cf5e620c	c6207
mind-live_3.12.30+23.06.23.01.62.iso	6703e425619a766ab521109885b51248	1d3ab04ace6895b042beb2d7ccfcd6c6cf5e620c	c6207
mind-live_3.12.30+23.06.23.01.63.iso	6703e425619a766ab521109885b51248	1d3ab04ace6895b042beb2d7ccfcd6c6cf5e620c	c6207
mind-live_3.12.30+23.06.23.01.70.iso	6703e425619a766ab521109885b51248	1d3ab04ace6895b042beb2d7ccfcd6c6cf5e620c	c6207
mind-live_3.12.30+23.06.23.01.80.iso	6703e425619a766ab521109885b51248	1d3ab04ace6895b042beb2d7ccfcd6c6cf5e620c	c6207
mind-live_3.12.30+23.06.23.01.90.iso	6703e425619a766ab521109885b51248	1d3ab04ace6895b042beb2d7ccfcd6c6cf5e620c	c6207
mind-live_3.12.30+23.06.23.01.95.iso	6703e425619a766ab521109885b51248	1d3ab04ace6895b042beb2d7ccfcd6c6cf5e620c	c6207
mind-live_3.12.31+23.07.23.01.30.iso	6703e425619a766ab521109885b51248	1d3ab04ace6895b042beb2d7ccfcd6c6cf5e620c	c6207

Network indicators

31.184.204.42
beacon.net.eu.org
c.glb-ru.info
claudfront.net
maxpatrol.net
nsdps.cc
rcsmf100.net
wmssh.com
dw-filter.com
net-sensors.net
mvs05.zyns.com

File signatures

```
rule PTESC_apt_multi_ZZ_DecoyDog__Trojan__FirstStage__v2{
        strings:
                $clefia_sbox = {57 49 D1 C6 2F 33 74 FB 95 6D 82 EA 0E B0 A8 1C}
                $linux = "machine-id"
                $windows1 = "\x00http\x00"
                $windows2 = "getaddrinfo"
                $windows3 = "VirtualProtect"
                $windows4 = "GetModuleFileNameA"
                $windows5 = "ReadFile"
        condition:
                ( uint16be ( 0 ) == 0x4d5a and all of ( $windows* ) or uint32be ( 0 )
== 0x7F454C46 and $linux ) and $clefia_sbox and filesize < 100KB
}

rule PTESC_tool_multi_ZZ_3snake__HackTool{
        strings:
                $a1 = "/proc/%d/cmdline"
                $a2 = "/proc/%d/status"
                $a3 = "/proc/%d/exe"
                $a4 = "/usr/bin/ssh"
                $a5 = "/usr/local/bin/"
                $a6 = "/usr/local/sbin/"
                $b1 = "intercept_ssh"
                $b2 = "sshd: [net]"
                $b3 = "sshd: [accepted]"
                $b4 = "[-] Plisteneter %d has been killed %d"
        condition:
                ( uint32be ( 0 ) == 0x7F454C46 ) and 4 of ( $a* ) and 2 of ( $b* )
}
```

MITRE TTPs

ID	Name	Description
		Resource Development
T1587.001	Develop Capabilities: Malware	The malicious actor develops its own attack tools based on open-source solutions for attack orchestration
		Initial Access
T1190	Exploit Public-Facing Application	The malicious actor compromises public-facing web services
T1199	Trusted Relationship	The malicious actor moves via adjacent infrastructures
T1078	Valid Accounts	The malicious actor used legitimate accounts to log in via SSH
T1021.004	Remote Services: SSH	The malicious actor connects to a compromised host over SSH
		Persistence
T1543.002	Create or Modify System Process: Systemd Service	Decoy Dog gained a foothold in the system using the dcrond or atd services
T1543.003	Create or Modify System Process: Windows Service	Decoy Dog gained a foothold in Windows-based infrastructure via system services
		Defense Evasion
T1480.001	Execution Guardrails: Environmental Keying	The malicious actor leveraged the victim host machine ID to encrypt a payload and configuration file
T1140	Deobfuscate/Decode Files or Information	The group encrypted its components with CLEFIA to protect these from detection and analysis
T1027.002	Obfuscated Files or Information: Software Packing	The group used a modified UPX algorithm as protection from detection and analysis
T1036.005	Masquerading: Match Legitimate Name or Location	The malicious actor used malware that copied the names of system utilities to avoid detection
		Credential Access
T1056	Input Capture	The malicious actor used a modified 3snake utility to intercept credentials
		Discovery
T1082	System Information Discovery	The malicious actor obtained the machine ID of an infected host to compile a Decoy Dog Loader that could run on the specific host only
		Command and Control
T1568.002	Dynamic Resolution: Domain Generation Algorithms	The group developed a domain generation algorithm (DGA)

ID	Name	Description
T1568.001	Dynamic Resolution: Fast Flux DNS	The group used DDNS services
T1071.004	Application Layer Protocol: DNS	DNS tunneling is the main method used by the Decoy Dog RAT for talking to its C2 server
		Impact
T1485	Data Destruction	The group destroyed Linux- and Windows-based infrastructure in an incident that involved a telecom company