Here’s a Simple Script to Detect the Stealthy Nation-State BPFDoor

blog.qualys.com/vulnerabilities-threat-research/2022/08/01/heres-a-simple-script-to-detect-the-stealthy-nation-state-bpfdoor

Harshal Tupsamudre August 1, 2022

```
int main(int argc, char *argv[])
{
        char hash[] = {0x6a, 0x75, 0x73, 0x74, 0x66, 0x6f, 0x72, 0x66, 0x75, 0x6e, 0x00}; // justforfun
        char hash2[]= {0x73, 0x6f, 0x63, 0x6b, 0x65, 0x74, 0x00}; // socket
        char *self[] = {
                "/sbin/udevd -d",
                "/sbin/mingetty /dev/tty7",
                "/usr/sbin/console-kit-daemon --no-daemon",
                "hald-addon-acpi: listening on acpi kernel interface /proc/acpi/event",
                "dbus-daemon --system",
                "hald-runner",
                "pickup -l -t fifo -u",
                "avahi-daemon: chroot helper",
                "/sbin/auditd -n",
                "/usr/lib/systemd/systemd-journald"
        };
```

In this blog, the Qualys Research Team explains the mechanics of a Linux malware variant named BPFdoor. We then demonstrate the efficacy of Qualys Custom Assessment and Remediation to detect it, and Qualys Multi-Vector EDR to protect against it.

BPFDoor is a Linux/Unix backdoor that allows threat actors to remotely connect to a Linux shell to gain complete access to a compromised device. It supports multiple protocols for communicating with a command & control server (C2) including TCP, UDP, and ICMP. It notably utilizes Berkeley Packet Filters (BPF) along with several other techniques to achieve these goals. BPF is a hooking function that allows a user-space program to attach a network filter onto any socket, and then allows or disallows certain types of data to come through that socket.

BPFDoor has been attributed to a Chinese threat actor group named Red Menshen (aka DecisiveArchitect), where the attackers have used it to gain stealthy remote access to compromised devices starting back in 2018 to the present day. Systems have been compromised across the US, South Korea, Hong Kong, Turkey, India, Vietnam, and Myanmar. Targets have included telecommunications, government, education, and logistics organizations. The group has been seen sending commands to BPFDoor victims via Virtual Private Servers (VPS) hosted at a well-known provider. In turn, these VPSs are administered via compromised routers based in Taiwan that the threat actor uses as VPN tunnels.

Target Geographies: Middle East, Asia

Target Sectors: Logistics, Education, Government

Malware Tools: Mangzamel, Gh0st, Gh0st, Metasploit, BPFDoor

Technical Analysis of BPFDoor

Execution

The threat actor leverages a custom implant tracked by the name “JustForFun”. When executed, the implant overwrites the process command line within the process environment by randomly selecting a new binary name from one of ten hard-coded options (shown in Figure 1). This masquerading technique is used to evade security solutions.

```
int main(int argc, char *argv[])
{
        char hash[] = {0x6a, 0x75, 0x73, 0x74, 0x66, 0x6f, 0x72, 0x66, 0x75, 0x6e, 0x00}; // justforfun
        char hash2[]= {0x73, 0x6f, 0x63, 0x6b, 0x65, 0x74, 0x00}; // socket
        char *self[] = {
                "/sbin/udevd -d",
                "/sbin/mingetty /dev/tty7",
                "/usr/sbin/console-kit-daemon --no-daemon",
                "hald-addon-acpi: listening on acpi kernel interface /proc/acpi/event",
                "dbus-daemon --system",
                "hald-runner",
                "pickup -l -t fifo -u",
                "avahi-daemon: chroot helper",
                "/sbin/auditd -n",
                "/usr/lib/systemd/systemd-journald"
        };
```

Figure 1: List of process names for Masquerading

The attacker interacts with the implant through the bash process to establish an interactive shell on a system. The command indicates the usage of Postfix queue manager (shown in Fig. 2).

```
qmgr -l -t fifo -u
```

```
int shell(int sock, char *rcmd, char *dcmd)
{
        int subshell;
        fd_set fds;
        char buf[BUF];
        char argx[] = {
                0x71, 0x6d, 0x67, 0x72, 0x20, 0x2d, 0x6c, 0x20, 0x2d, 0x74,
                0x20, 0x66, 0x69, 0x66, 0x6f, 0x20, 0x2d, 0x75, 0x00}; // qmgr -l -t fifo -u
        char *argvv[] = {argx, NULL, NULL};
        #define MAXENV 256
        #define ENVLEN 256
        char *envp[MAXENV];
        char sh[] = {0x2f, 0x62, 0x69, 0x6e, 0x2f, 0x73, 0x68, 0x00}; // /bin/sh
```

Figure 2: Encoded shell and qmgr commands

Masquerading (Rename the process)

```
int set_proc_name(int argc, char **argv, char *new)
{
        size_t size = 0;
        int i;
        char *raw = NULL;
        char *last = NULL;

        argv0 = argv[0];

        for (i = 0; environ[i]; i++)
                size += strlen(environ[i]) + 1;

        raw = (char *) malloc(size);
        if (NULL == raw)
                return -1;

        for (i = 0; environ[i]; i++)
        {
                memcpy(raw, environ[i], strlen(environ[i]) + 1);
                environ[i] = raw;
                raw += strlen(environ[i]) + 1;
        }

        last = argv[0];

        for (i = 0; i < argc; i++)
                last += strlen(argv[i]) + 1;
        for (i = 0; environ[i]; i++)
                last += strlen(environ[i]) + 1;

        memset(argv0, 0x00, last - argv0);
        strncpy(argv0, new, last - argv0);

        prctl(PR_SET_NAME, (unsigned long) new);
        return 0;
}
```

Figure 3: Code uses prctl to rename the malware process

The malware will rename itself using the prctl function with the argument PR_SET_NAME, and a random legitimate-looking name (Fig. 3). These names are hardcoded in the binary and vary between the samples.

Timestomping

```
static void setup_time(char *file)
{
        struct timeval tv[2];

        tv[0].tv_sec = 1225394236;
        tv[0].tv_usec = 0;

        tv[1].tv_sec = 1225394236;
        tv[1].tv_usec = 0;

        utimes(file, tv);
}
```

Figure 4: Code for Timestomping

The implant sets a fake time to timestomp the binary before deletion. A function dubbed set_time was called to alter the access and modification timestamp of the binary using the utimes function (Fig. 4). The timestamp used was always set to Thursday, October 30, 2008 7:17:16 PM (GMT).

PID File

The implant creates a zero-byte PID file at /var/run/haldrund.pid (Fig. 5). The file has two conditions:

- This file is deleted if the implant terminates normally,
- The file is not deleted, if there is a problem like hard shutdown or crash.

The implant will not resume if this file is present as it describes the running state for the backdoor.

```
pid_path[0] = 0x2f; pid_path[1] = 0x76; pid_path[2] = 0x61;
pid_path[3] = 0x72; pid_path[4] = 0x2f; pid_path[5] = 0x72;
pid_path[6] = 0x75; pid_path[7] = 0x6e; pid_path[8] = 0x2f;
pid_path[9] = 0x68; pid_path[10] = 0x61; pid_path[11] = 0x6c;
pid_path[12] = 0x64; pid_path[13] = 0x72; pid_path[14] = 0x75;
pid_path[15] = 0x6e; pid_path[16] = 0x64; pid_path[17] = 0x2e;
pid_path[18] = 0x70; pid_path[19] = 0x69; pid_path[20] = 0x64;
pid_path[21] = 0x00; // /var/run/haldrund.pid
```

Figure 5: Encoded command for creating PID file

BPFDoor Detection using Qualys Custom Assessment & Remediation

Qualys Custom Assessment and Remediation can be leveraged to create and execute custom detection logics for zero-day threats. This cloud service supports multiple scripting languages including Perl, Shell, Python, Lua, PowerShell, and VBScript with no vendor-

specific syntax or restrictions. Select the language of your choice and start by leveraging out-of-the-box scripts or creating your own scripts for custom detection, validation, and remediation.

We created the Shell script as part of our detection logic via the Qualys scripting service and executed it across the network.

Using this script, we are looking for packet sniffing processes under the entire process stack and checking if an existing process has opened a raw socket using the default Linux utility lsof. Refer the following screenshots of the script (Fig. 6) and its output (Fig. 7).

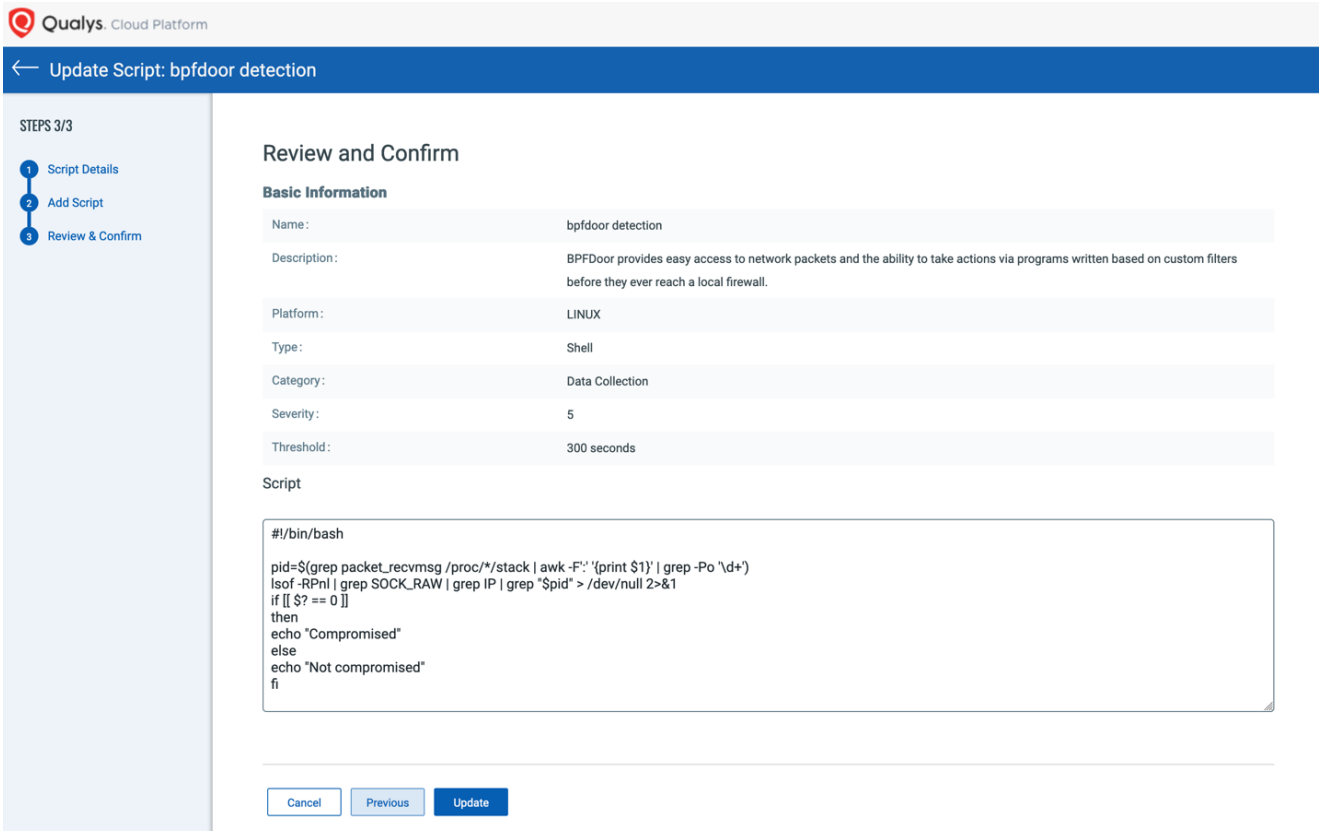

Figure 6: Script to detect BPFdoor

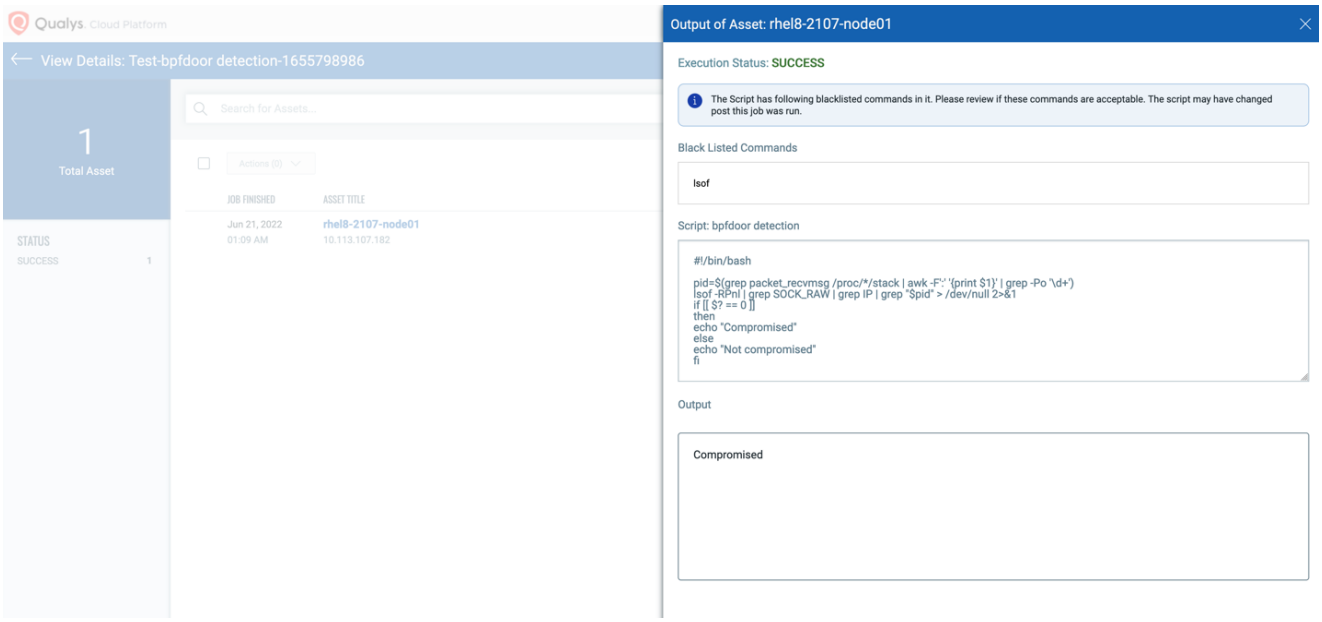

Figure 7: BPFdoor detection

BPFDoor Detection using Qualys Multi-Vector EDR

Qualys Multi-Vector EDR, armed with YARA scanning techniques, detects the BPFdoor RAT with a threat score of 5/10 (Fig. 8).

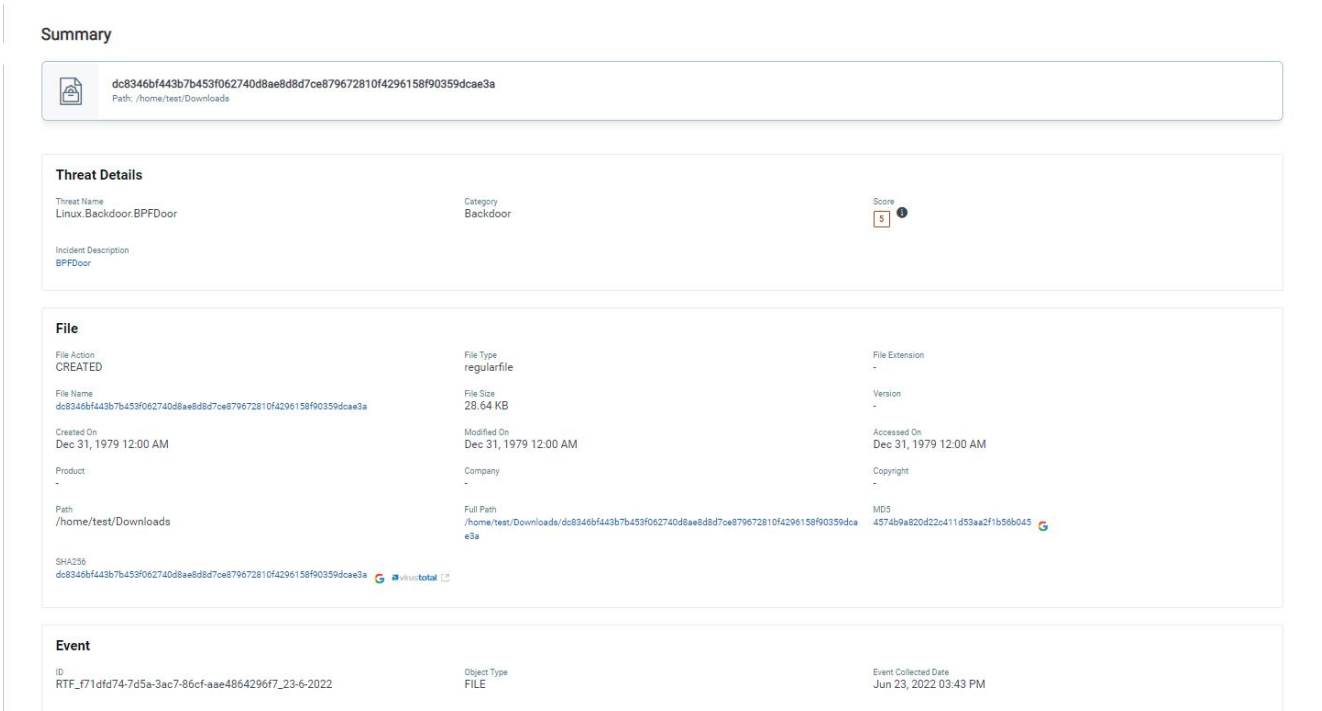

Figure 8: Qualys Multi-Vector EDR detection for BPFdoor

After execution, the binary masquerades its name by selecting from one of 10 names randomly:

```
/sbin/udevd -d
/sbin/mingetty /dev/tty7
/usr/sbin/console-kit-daemon --no-daemon
hald-addon-acpi: listening on acpi kernel interface /proc/acpi/event
dbus-daemon --system
hald-runner
pickup -l -t fifo -u
avahi-daemon: chroot helper
/sbin/auditd -n
/usr/lib/systemd/systemd-journald
```

The highlighted name was used during the execution. The names are made to look like common Linux system daemons. The implant overwrites the argv[0] value which is used by the Linux /proc filesystem to determine the command line and command name to show for each process. By doing this, when a run command like ps is executed, it shows the fake name.

The renamed binary is dropped to the /dev/shm directory and runs itself as /dev/shm/kdmtmpflush (Figs. 9 and 10). The masqueraded process with a “–init” flag tells itself to execute secondary clean-up operations and go resident.

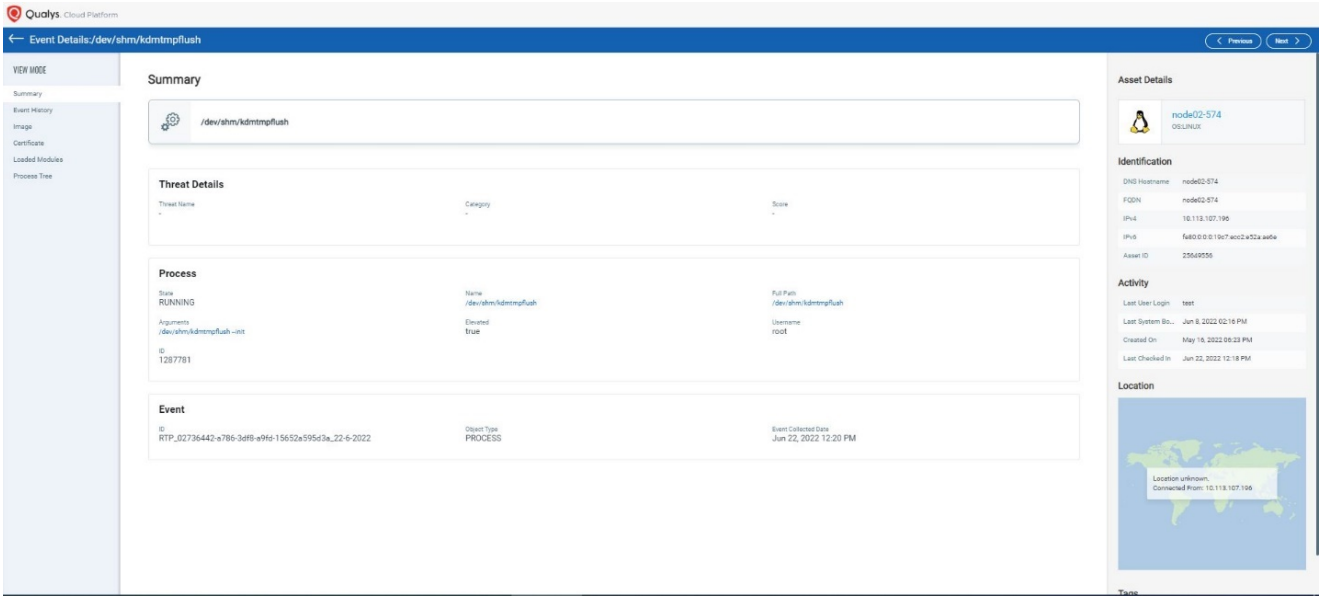

Figure 9: Qualys Multi-Vector EDR telemetry for detecting Masquerading

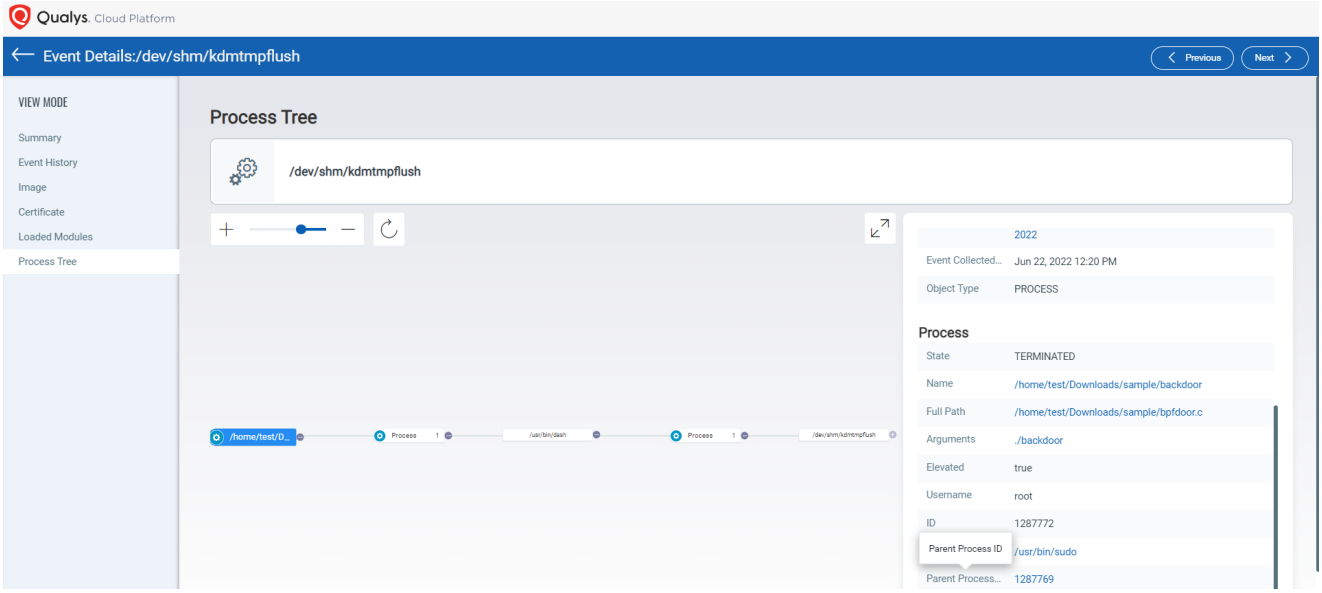

Figure 10: BPFdoor Process Tree

The implant creates a zero-byte PID file at `/var/run/haldrund.pid` (Fig. 11).

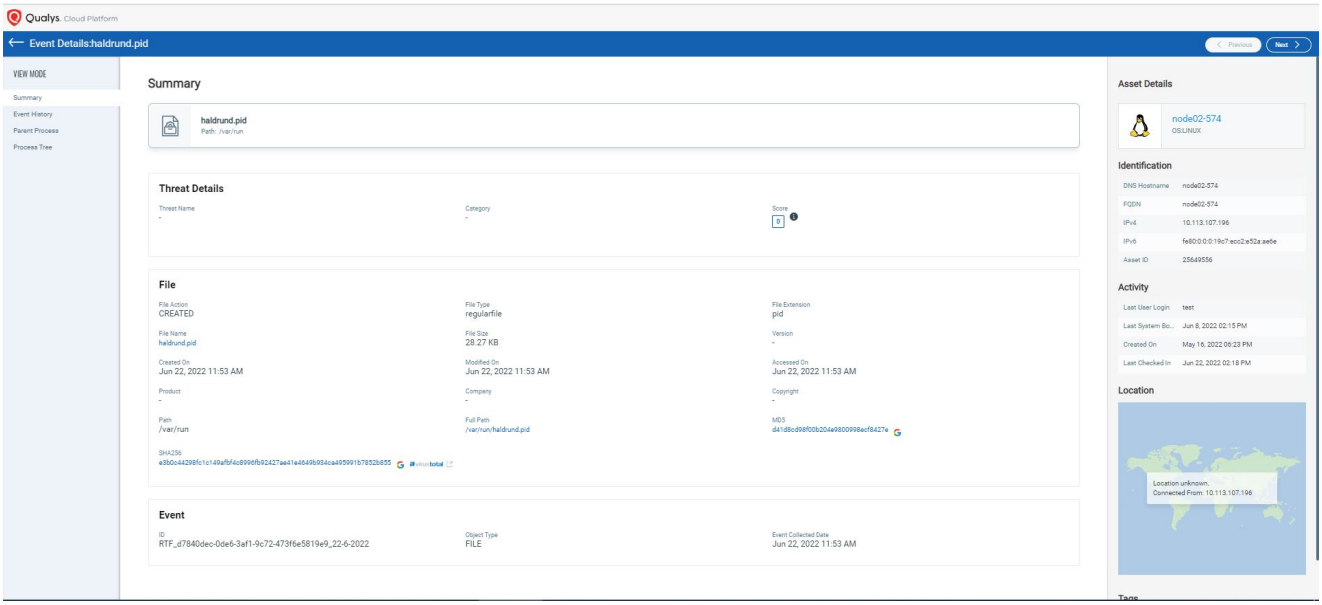

Figure 11: Creation of PID file by BPFdoor

As shown in figure 12, The original execution process deletes `/dev/shm/kdmtmpflush` with the following command:

```
/bin/rm -f /dev/sfm/kdmtmpflush
```

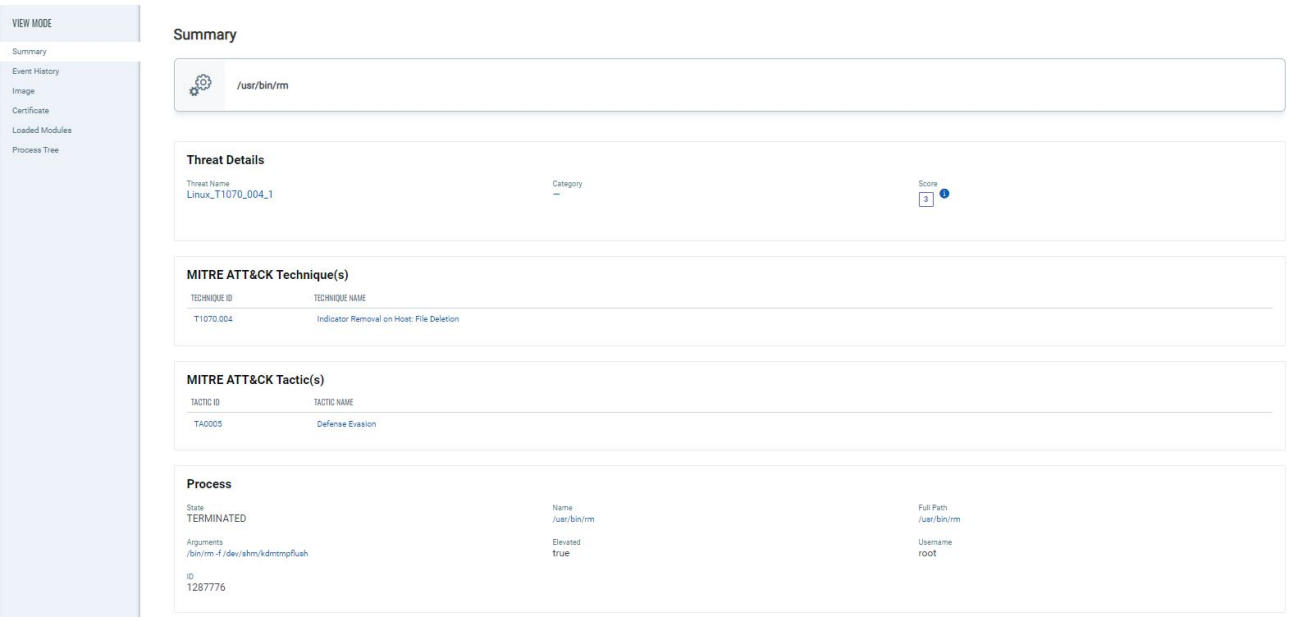

Figure 12: Deletion of /dev/shm/kdmtmpflush directory

Conclusion

As with most remote access tools, BPFDoor is visible during the post-exploitation phase of an attack. It is expected that the authors behind BPFdoor will be upgrading its functionality over time, including different commands, processes, or files. This malware has a vast arsenal at its disposal. Therefore, we recommend that organizations have a robust EDR solution to both detect its signatures and adequately respond to the threat.

MITRE ATT&CK Techniques

T1036.005- Masquerading: Match Legitimate Name or Location
T1070.004- Indicator Removal on Host: File Deletion
T1070.006- Indicator Removal on Host: Time Stomp
T1059.004- Command and Scripting Interpreter: Unix Shell
T1106- Native API
T1548.001- Abuse Elevation Control Mechanism: Setuid and Setgid
T1095- Non-Application Layer Protocol

IoC (Indicators of Compromise)

Hashes (SHA256)

07ecb1f2d9ffbd20a46cd36cd06b022db3cc8e45b1ecab62cd11f9ca7a26ab6d
1925e3cd8a1b0bba0d297830636cdb9ebf002698c8fa71e0063581204f4e8345
4c5cf8f977fc7c368a8e095700a44be36c8332462c0b1e41bff03238b2bf2a2d
591198c234416c6ccbcea6967963ca2ca0f17050be7eed1602198308d9127c78
599ae527f10ddb4625687748b7d3734ee51673b664f2e5d0346e64f85e185683
5b2a079690efb5f4e0944353dd883303ffd6bab4aad1f0c88b49a76ddcb28ee9

5faab159397964e630c4156f8852bcc6ee46df1cdd8be2a8d3f3d8e5980f3bb3
76bf736b25d5c9aaf6a84edd4e615796fffc338a893b49c120c0b4941ce37925
93f4262fce8c6b4f8e239c35a0679fbbbb722141b95a5f2af53a2bcafe4edd1c
96e906128095dead57fdc9ce8688bb889166b67c9a1b8fdb93d7cff7f3836bb9
97a546c7d08ad34dfab74c9c8a96986c54768c592a8dae521ddcf612a84fb8cc
c796fc66b655f6107eacbe78a37f0e8a2926f01fecebd9e68a66f0e261f91276
c80bd1c4a796b4d3944a097e96f384c85687daeedcdcf05cc885c8c9b279b09c
f47de978da1dbfc5e0f195745e3368d3ceef034e964817c66ba01396a1953d72
f8a5e735d6e79eb587954a371515a82a15883cf2eda9d7ddb8938b86e714ea27
fa0defdabd9fd43fe2ef1ec33574ea1af1290bd3d763fdb2bed443f2bd996d73
fd1b20ee5bd429046d3c04e9c675c41e9095bea70e0329bd32d7edd17ebaf68a
144526d30ae747982079d5d340d1ff116a7963aba2e3ed589e7ebc297ba0c1b3
fa0defdabd9fd43fe2ef1ec33574ea1af1290bd3d763fdb2bed443f2bd996d73
76bf736b25d5c9aaf6a84edd4e615796fffc338a893b49c120c0b4941ce37925
96e906128095dead57fdc9ce8688bb889166b67c9a1b8fdb93d7cff7f3836bb9
c80bd1c4a796b4d3944a097e96f384c85687daeedcdcf05cc885c8c9b279b09c
f47de978da1dbfc5e0f195745e3368d3ceef034e964817c66ba01396a1953d72
07ecb1f2d9ffbd20a46cd36cd06b022db3cc8e45b1ecab62cd11f9ca7a26ab6d
4c5cf8f977fc7c368a8e095700a44be36c8332462c0b1e41bff03238b2bf2a2d
599ae527f10ddb4625687748b7d3734ee51673b664f2e5d0346e64f85e185683
5b2a079690efb5f4e0944353dd883303ffd6bab4aad1f0c88b49a76ddcb28ee9
5faab159397964e630c4156f8852bcc6ee46df1cdd8be2a8d3f3d8e5980f3bb3
93f4262fce8c6b4f8e239c35a0679fbbbb722141b95a5f2af53a2bcafe4edd1c
97a546c7d08ad34dfab74c9c8a96986c54768c592a8dae521ddcf612a84fb8cc
c796fc66b655f6107eacbe78a37f0e8a2926f01fecebd9e68a66f0e261f91276
f8a5e735d6e79eb587954a371515a82a15883cf2eda9d7ddb8938b86e714ea27
fd1b20ee5bd429046d3c04e9c675c41e9095bea70e0329bd32d7edd17ebaf68a

Filenames

```
/dev/shm/kdmtmpflush
/dev/shm/kdumpflush
/dev/shm/kdumpdb
/var/run/xinetd.lock
/var/run/kdevrund.pid
/var/run/haldrund.pid
/var/run/syslogd.reboot
```

Process names

```
/sbin/udevd -d
/sbin/mingetty /dev/tty7
/usr/sbin/console-kit-daemon –no-daemon
hald-addon-acpi: listening on acpi kernel interface /proc/acpi/event
dbus-daemon –system
hald-runner
pickup -l -t fifo -u
avahi-daemon: chroot helper
/sbin/auditd -n
/usr/lib/systemd/systemd-journald
/usr/libexec/postfix/master
qmgr -l -t fifo -u
```

Contributors:

Viren Chaudhary (Senior Engineer, Threat Research, Qualys)
Mukesh Choudhary (Compliance Research Analyst, Qualys)
Lavish Jhamb (Solutions Architect, Compliance Solutions, Qualys)
Mohd Anas Khan (Compliance Research Analyst, Qualys)