OMNI (MIRAI variant) Botnet Infections Detected in Polycom Video Conferencing Devices

Report

Overview

Mirai botnet has been used to infect IoT devices at large scale to perform unauthorized operations. Mirai botnet is formed by exploiting vulnerabilities, weak configurations, insecure authentication, etc. in the IoT devices. The compromised IoT devices are then used to nefarious operations such as launch Brute-force attacks, Distributed Denial of Service (DDoS) attacks, turning IoT device into a proxy server and others. Mirai botnet was the major source of launching DDoS against Dyn DNS solution provider . In this research note, WITL detected OMNI (Mirai variant) infected PolyCom video conferencing devices during in-house research and the devices are still infected with this botnet on the Internet.

The research was targeted against PolyCom HDX devices.

Analysis

It has been found that the PolyCom devices are shipped with BusyBox, Wget, and other binaries. Mirai (and variants) extensively use the binaries such as BusyBox, Wget, and other for performing different set of operations. OMNI (Mirai Botnet) has been used to infect GPON routers using CVE-2018-10561 vulnerability. However, WITL noticed OMNI infections in the PolyCom HDX systems.

BusyBox is considered as Swiss Army knife for embedded systems. As per the BusyBox documentation - *BusyBox combines tiny versions of many common UNIX utilities into a single small executable. It provides replacements for most of the utilities you usually find in GNU fileutils, shellutils, etc. The utilities in BusyBox generally have fewer options than their full-featured GNU cousins; however, the options that are included provide the expected functionality and behave very much like their GNU counterparts. BusyBox provides a fairly complete environment for any small or embedded system.*

OMNI bot recon phase with brute-force attack execution (DEBUG logs generated from the Polycom device) is discussed below:

1. The OMNI bot triggers the “enable” command
2. After that, “system” command is executed
3. Once the “system” command is executed, the “sh” command is triggered to obtain the shell access so that specific binary can be executed
4. After entering the shell with “sh”, The OMNI bot calls its main functions by loading the “/bin/busybox” with OMNI as “/bin/busybox/ OMNI”

The communication is highlighted below:

```
2018-07-01 14:49:02 DEBUG avc: pc[0]: uimsg: C: enable
2018-07-01 14:49:02 DEBUG avc: pc[0]: appcom: api_command: pid:1181 2 enable
2018-07-01 14:49:03 DEBUG avc: pc[0]: appcom: legacy_api_command: 2 enable 1
2018-07-01 14:49:03 DEBUG avc: pc[0]: api: legacy_api_c_command: enable
2018-07-01 14:49:03 DEBUG avc: pc[0]: standardCmndHandler(...)
2018-07-01 14:49:03 DEBUG avc: pc[0]: legacy_api: legacy_api_c_command succeeded


2018-07-01 14:49:04 DEBUG avc: pc[0]: uimsg: C: system
2018-07-01 14:49:04 DEBUG avc: pc[0]: appcom: api_command: pid:1181 2 system
2018-07-01 14:49:04 DEBUG avc: pc[0]: appcom: legacy_api_command: 2 system 1
2018-07-01 14:49:04 DEBUG avc: pc[0]: appcom: java_api_command: pid:1181 2 system
2018-07-01 14:49:04 DEBUG avc: pc[0]: "system" - takes at least one argument


2018-07-01 14:49:05 DEBUG avc: pc[0]: uimsg: C: sh
2018-07-01 14:49:05 DEBUG avc: pc[0]: appcom: api_command: pid:1181 2 sh
2018-07-01 14:49:05 DEBUG avc: pc[0]: appcom: legacy_api_command: 2 sh 1
2018-07-01 14:49:06 DEBUG avc: pc[0]: appcom: java_api_command: pid:1181 2 sh

2018-07-01 14:49:06 DEBUG avc: pc[0]: Usage: sh_button <"button" <up|down> | help>
2018-07-01 14:49:06 DEBUG avc: pc[0]: uimsg: C: /bin/busybox OMNI
2018-07-01 14:49:06 DEBUG avc: pc[0]: appcom: api_command: pid:1181 2 /bin/busybox OMNI
2018-07-01 14:49:06 DEBUG avc: pc[0]: appcom: legacy_api_command: 2 /bin/busybox OMNI 1
2018-07-01 14:49:06 DEBUG avc: pc[0]: appcom: java_api_command: pid:1181 2 /bin/busybox OMNI

2018-07-01 14:49:09 INFO avc: pc[0]: config: synching filesystem
2018-07-01 14:49:09 INFO avc: pc[0]: config: sync complete 327ms
2018-07-01 14:49:40 DEBUG avc: pc[0]: XCOM:WARNING:remote socket closed

2018-07-01 14:49:40 WARNING avc: pc[0]: appcom: psh session closed socket abruptly
2018-07-01 14:49:40 DEBUG avc: pc[0]: appcom: unregister_api_session 2
2018-07-01 14:49:41 DEBUG avc: pc[0]: uimsg: R: telnet /tmp/apiasynclisteners/psh0
2018-07-01 14:49:41 DEBUG avc: pc[0]: appcom: allocate_session(fd=106)
2018-07-01 14:49:41 DEBUG avc: pc[0]: appcom: register_api_session new_session_p=675763d8
2018-07-01 14:49:41 DEBUG avc: pc[0]: appcom: add_session(session_p:675763d8)
2018-07-01 14:49:41 DEBUG avc: pc[0]: appcom: about to call sendJavaMessageEx
2018-07-01 14:49:41 DEBUG jvm: pc[0]: UI: xcom-api: ClientManager: createSession(type: telnet sess: 2
2018-07-01 14:49:41 DEBUG avc: pc[0]: appcom: session 2 registered
```

5. Once the primary "/bin/busybox/ OMNI" is loaded, it triggers the brute-force attacks as explained below:
    a. A command is executed as "execwithoutecho systemsetting telnet_client_"
        i. The "execwithoutecho" executes the command without throwing any notification or output via echo.
        ii. The command executed is "telnet_client" which opens connection to remote IP running telnet service
        iii. The "OMNI" bot then uses combination of username and password as strings. In the DEBUG logs presented below, it started with "root:root", "zlxx:zlxx"

The communication logs are shown below:

```
2018-07-01 14:49:41 DEBUG avc: pc[0]: uimsg: E: execwithoutecho systemsetting telnet_client_23 188.18.131.63
2018-07-01 14:49:41 DEBUG avc: pc[0]: appcom: legacy_api_command: 2 execwithoutecho systemsetting
telnet_client_23 188.18.131.63 1
```

```
2018-07-01 14:49:41 DEBUG avc: pc[0]: appcom: java_api_command: pid:1181 2 execwithoutecho systemsetting
telnet_client_23 188.18.131.63

2018-07-01 14:49:41 DEBUG avc: pc[0]: appcom: java_api_command succeeded (sess:2)
2018-07-01 14:49:41 DEBUG avc: pc[0]: uimsg: E: execwithoutecho systemsetting telnet_client_23_success True
2018-07-01 14:49:41 DEBUG avc: pc[0]: appcom: legacy_api_command: 2 execwithoutecho systemsetting
telnet_client_23_success True 1
2018-07-01 14:49:41 DEBUG avc: pc[0]: appcom: java_api_command: pid:1181 2 execwithoutecho systemsetting
telnet_client_23_success True
2018-07-01 14:49:41 DEBUG avc: pc[0]: appcom: java_api_command succeeded (sess:2)

2018-07-01 14:49:41 DEBUG avc: pc[0]: uimsg: C: root
2018-07-01 14:49:41 DEBUG avc: pc[0]: appcom: api_command: pid:1181 2 root
2018-07-01 14:49:41 DEBUG avc: pc[0]: appcom: legacy_api_command: 2 root 1
2018-07-01 14:49:41 DEBUG avc: pc[0]: appcom: java_api_command: pid:1181 2 root


2018-07-01 14:49:42 DEBUG avc: pc[0]: uimsg: C: zlxx.
2018-07-01 14:49:42 DEBUG avc: pc[0]: appcom: api_command: pid:1181 2 zlxx.
2018-07-01 14:49:42 DEBUG avc: pc[0]: appcom: legacy_api_command: 2 zlxx. 1
2018-07-01 14:49:42 DEBUG avc: pc[0]: appcom: java_api_command: pid:1181 2 zlxx.
```

After trying few combinations of username and password, the process is repeated again as shown below:

```
2018-07-01 14:49:43 DEBUG avc: pc[0]: uimsg: C: enable
2018-07-01 14:49:43 DEBUG avc: pc[0]: appcom: api_command: pid:1181 2 enable
2018-07-01 14:49:43 DEBUG avc: pc[0]: appcom: legacy_api_command: 2 enable 1
2018-07-01 14:49:43 DEBUG avc: pc[0]: api: legacy_api_c_command: enable
2018-07-01 14:49:43 DEBUG avc: pc[0]: standardCmndHandler(...)
2018-07-01 14:49:43 DEBUG avc: pc[0]: legacy_api: legacy_api_c_command succeeded

2018-07-01 14:49:43 DEBUG avc: pc[0]: uimsg: C: system
2018-07-01 14:49:43 DEBUG avc: pc[0]: appcom: api_command: pid:1181 2 system
2018-07-01 14:49:43 DEBUG avc: pc[0]: appcom: legacy_api_command: 2 system 1
2018-07-01 14:49:43 DEBUG avc: pc[0]: appcom: java_api_command: pid:1181 2 system
2018-07-01 14:49:43 DEBUG avc: pc[0]: "system" - takes at least one argument

2018-07-01 14:49:45 DEBUG avc: pc[0]: uimsg: C: sh
2018-07-01 14:49:45 DEBUG avc: pc[0]: appcom: api_command: pid:1181 2 sh
2018-07-01 14:49:45 DEBUG avc: pc[0]: appcom: legacy_api_command: 2 sh 1
2018-07-01 14:49:45 DEBUG avc: pc[0]: appcom: java_api_command: pid:1181 2 sh
2018-07-01 14:49:45 DEBUG avc: pc[0]: Usage: sh_button <"button" <up|down> | help>

2018-07-01 14:49:45 DEBUG avc: pc[0]: uimsg: C: /bin/busybox OMNI
2018-07-01 14:49:45 DEBUG avc: pc[0]: appcom: api_command: pid:1181 2 /bin/busybox OMNI
2018-07-01 14:49:45 DEBUG avc: pc[0]: appcom: legacy_api_command: 2 /bin/busybox OMNI 1
2018-07-01 14:49:45 DEBUG avc: pc[0]: appcom: java_api_command: pid:1181 2 /bin/busybox OMNI
2018-07-01 14:50:16 DEBUG avc: pc[0]: uimsg: E: execwithoutecho systemsetting telnet_client_23

2018-07-01 14:50:57 DEBUG avc: pc[0]: appcom: legacy_api_command: 2 execwithoutecho systemsetting
telnet_client_23 188.18.131.63 1
2018-07-01 14:50:57 DEBUG avc: pc[0]: appcom: java_api_command: pid:1181 2 execwithoutecho systemsetting
telnet_client_23 188.18.131.63
2018-07-01 14:50:57 DEBUG avc: pc[0]: appcom: java_api_command succeeded (sess:2)
2018-07-01 14:50:57 DEBUG avc: pc[0]: uimsg: E: execwithoutecho systemsetting telnet_client_23_success True
2018-07-01 14:50:57 DEBUG avc: pc[0]: appcom: legacy_api_command: 2 execwithoutecho systemsetting
telnet_client_23_success True 1
2018-07-01 14:50:57 DEBUG avc: pc[0]: appcom: java_api_command: pid:1181 2 execwithoutecho systemsetting
```

```
telnet_client_23_success True
2018-07-01 14:50:57 DEBUG avc: pc[0]: appcom: java_api_command succeeded (sess:2)
2018-07-01 14:50:57 DEBUG avc: pc[0]: uimsg: C: adm
2018-07-01 14:50:57 DEBUG avc: pc[0]: appcom: api_command: pid:1181 2 adm
2018-07-01 14:50:57 DEBUG avc: pc[0]: appcom: legacy_api_command: 2 adm 1
2018-07-01 14:50:57 DEBUG avc: pc[0]: appcom: java_api_command: pid:1181 2 adm


-- [Truncated] ---
```

If the DEBUG logs are dissected in detail, it can be noticed that the "appcom:ap_command", "appcom:legacy_api_command", "appcom:java_api_command" are primarily belong to the AVC binary. A number of artefacts related to SVC/AVC are discussed below:

- H.264 Advanced Video Coding (AVC), an efficient and high-performance standard that is used by most of today's video conferencing devices.
- H.264 Scalable Video Coding (SVC) is an extension to H.264 Advanced Video Coding (AVC)
- Scalable Video Coding (SVC) is a newer form of video compression which dynamically adjusts the frame rate or resolution in real-time based on varying network conditions
- SVC is preferred to deploy a soft video client for entire organization on a shared network that is shared by many other applications. However, AVC is preferred for a more controlled network environment where QoS can be implemented to ensure a time-sensitive data
- SVC/AVC provides an API interface or related functions that are utilized by attackers to execute certain commands.

Inferences

- Polycom HDX devices for audio/video conferencing have been found to be infected with OMNI botnet, a variant of MIRAI
- The attackers are harnessing the power of open-source software packages such as "BusyBox", WGet", and others that are shipped with the embedded firmware of the Polycom devices.
- Compromised Polycom devices are used to launch brute-force attacks, potential DDoS attacks and also been used as proxy devices for routing malicious communications such as Command and Control (C&C).
- APIs supported by Polycom are abused by the attackers for performing operations in the device