AKAMAI THREAT ADVISORY

SSHowDowN

Exploitation of IoT devices for Launching Mass-Scale Attack Campaigns

By Ezra Caltum & Ory Segal, Akamai Threat Research

Akamai
FASTER FORWARD

Issue Date: 10.11.16

Executive Summary

How many times do end users think about the factory default settings of their Internet-connected devices? Perhaps we all should. The Akamai's Threat Research team recently reported on a case where millions of Internet-connected (IoT) devices were being used as the source for web based credential stuffing campaigns. When we dug a little deeper, we found evidence that these IoT devices were being used as proxies to route malicious traffic due to some default configuration weaknesses in their operating systems.

While this has been reported before, the vulnerability has resurfaced with the increase of connected devices. Our team is currently working with the most prevalent device vendors on a proposed plan of mitigation. We would like to emphasize that this is not a new type of vulnerability or attack technique, but rather a weakness in many default configurations of Internet-connected devices, which is actively being exploited in mass scale attack campaigns against Akamai customers.

We observed SSHowDowN Proxy attacks from the following types of devices, and other devices types are likely vulnerable as well.

- CCTV, NVR, DVR devices (video surveillance)
- Satellite antenna equipment
- Networking devices (e.g. Routers, Hotspots, WiMax, Cable and ADSL modems, etc.)
- Internet connected NAS devices (Network Attached Storage)

Vulnerable connected devices are being used for:

1. Mounting attacks against any kind of Internet target and against any kind of Internet-facing service such as HTTP, SMTP and Network Scanning

2. Mounting attacks against internal networks that host these connected devices

Once malicious users access the web administration console of these devices they can then compromise the device's data and in some cases, take over the machine.

In this case, unauthorized SSH tunnels were created and used, despite the fact that the IoT devices were supposedly hardened and do not allow the default web interface user to SSH into the device and execute commands. Due to this, we feel compelled to reiterate the warning.

How to Protect Yourself

End users:

1. **Always** change factory-default credentials of any Internet-connected device
2. Unless required for normal operation, completely disable the SSH service on any Internet-connected device. If SSH is required, put "`AllowTcpForwarding No`" into sshd_config.
3. Consider establishing inbound firewall rules preventing SSH access to your IOT devices from outside of a narrowly trusted IP space, such as your own internal network.
4. Consider establishing outbound firewall rules in place for IOT devices at your network boundary, preventing tunnels established from resulting in successful outbound connections.

Device vendors:

1. Avoid shipping Internet-connected devices with undocumented accounts
2. Disable SSH on devices unless absolutely required for normal operations
3. Force users to change factory default account credentials after initial installation
4. Configure SSH to disallow TCP Forwarding
5. Provide a secure process for end-users to update sshd configuration so that they may mitigate future vulnerabilities without having to wait for a firmware patch.

Technical Details / Recently, Akamai's Threat Research Team, and other multiple security vendors and research teams, reported on a trend where IoT devices are being exploited in order to mount attacks against third party victims. These devices were leveraged to conduct a mass-scale HTTP-based credential stuffing campaigns against customers.

We would like to emphasize that this is not a new type of vulnerability or attack technique, but rather a weakness in many default configurations of IoT devices. In fact, several articles were previously released, all touching similar topics. For example:

- A *blog post* by Brian Krebs ("IoT Reality: Smart Devices, Dumb Defaults")
- An *article* by Jeff Huckaby, which describes how hackers are using SSH tunnels to send spam
- *CVE-2004-1653*, which was published against OpenSSH for allowing TCP forwarding by default, and the risk this causes for service accounts designed to not allow normal shell access
- Jordan Sissel wrote *an article discussing the dangers of using /bin/false*
- Joey Hess discusses the insecurity involved in the default SSH TCP forwarding configuration in his *blog*

After analyzing large data sets from Akamai's Cloud Security Intelligence platform, we discovered several common features, which led us to believe that the IoT devices were being used as proxies to route malicious traffic against victim sites.

In order to prove our hypothesis, we acquired and installed identical devices that were used in the attacks, in a connected threat research lab, and decided to work to uncover the root cause and techniques used by the attackers, in order to find out how we can better protect ourselves, our customers and all IoT device users.

Forensic Analysis / After seeing suspicious traffic against Akamai customers that originated from a Network Video Recorder (*NVR*) device, we checked if any unauthorized users were logged into the machine and running commands. This check showed no active users except for ourselves.

We then wanted to check what processes were responsible for the active malicious HTTP/HTTPS connections. Since the UNIX "netstat" utility was not available on this device type, we used the "ss" command (a *utility* for investigating sockets) to list the live network connections and the PIDs (process IDs) related to them:

```
 users:(("sshd",pid=881,fd=8))                                    10.1.10.201:35981          [redacted]:http
tcp    CLOSE-WAIT 0      0
 users:(("sshd",pid=881,fd=24))                                   10.1.10.201:ssh            [redacted]:58087
tcp    ESTAB      0      0
 users:(("sshd",pid=4117,fd=3),("sshd",pid=4109,fd=3))            10.1.10.201:51169          [redacted]:https
tcp    FIN-WAIT-2 0      0
 users:(("sshd",pid=14651,fd=24))                                 10.1.10.201:47285          [redacted]:http
tcp    ESTAB      0      0
 users:(("sshd",pid=8723,fd=7))                                   10.1.10.201:46327          [redacted]:http
tcp    ESTAB      0      0
 users:(("sshd",pid=8746,fd=14))                                  10.1.10.201:48656          [redacted]:https
tcp    FIN-WAIT-2 0      0
 users:(("sshd",pid=14651,fd=32))                                 10.1.10.201:47470          [redacted]:https
tcp    ESTAB      0      0
 users:(("sshd",pid=387,fd=15))                                   10.1.10.201:36069          [redacted]:https
tcp    ESTAB      0      0
 users:(("sshd",pid=8691,fd=20))                                  10.1.10.201:57375          [redacted]:https
tcp    FIN-WAIT-2 0      0
 users:(("sshd",pid=14651,fd=55))                                 10.1.10.201:ssh            [redacted]:60348
tcp    ESTAB      0      100
 users:(("sshd",pid=881,fd=3),("sshd",pid=872,fd=3))              10.1.10.201:49975          [redacted]:https
tcp    ESTAB      0      0
 users:(("sshd",pid=4391,fd=13))                                  10.1.10.201:ssh            [redacted]:58674
tcp    ESTAB      0      0
```

A quick look showed that the SSH daemon (sshd) was the process responsible for all active HTTP/HTTPS connections.

The next thing we wanted to find out was which user was running these sshd processes. In our case, the processes were executed by the user called "admin". This user is a factory-default administrator, that is shipped with this brand of NVR system. The vendor's documentation also revealed that this user is configured with the default password "admin".

We checked our factory-defaulted device and noticed that the "admin:admin" credential pair allows us to connect to the web-based configuration interface. However, the default configuration does not allow the "admin" user to connect to the device via SSH – as shown in the /etc/passwd file (the "command/shell" field is set to /sbin/nologin):

```
root:x:0:0:root:/root:/bin/bash
nobody:x:99:99:Unprivileged User:/dev/null:/bin/false
sshd:x:50:50:sshd PrivSep:/var/lib/sshd:/bin/false
ftp:x:45:45:anonymous_user:/home/ftp:/bin/false
messagebus:x:18:18:D-BUS Message Daemon User:/dev/null:/bin/false
admin:x:600:600::/var /sbin/nologin
localdisplay:x:700:700::/tmp:/sbin/nologin
```

Any remote user attempting to SSH into the machine and to execute commands, would be greeted with the message "***This account is currently not available***" and then immediately disconnected from the system. This has been the standard method to date for having system users which are unable to log in directly.

At this point, we discovered the following facts:

- A remote user was able to cause the device to generate HTTP traffic
- The remote user was using the device's SSH daemon
- The execution of the SSH daemon was under the factory-default "admin" user
- There was no active shell session for the "admin" user, which makes sense given that the "admin" user has the "/sbin/nologin" configured as the shell

NVR Device Exploitation / We quickly connected the dots and speculated that an attacker might be leveraging SSH's capability to serve as a SOCKS proxy by using the **-D** option, which the SSH man page describes as follows:

> *"[-D] Specifies a local dynamic application-level port forwarding... Whenever a connection is made to this port, the connection is forwarded over the secure channel, and the application protocol is then used to determine where to connect to from the remote machine. Currently the SOCKS4 and SOCKS5 protocols are supported, and SSH will act as a SOCKS server"*

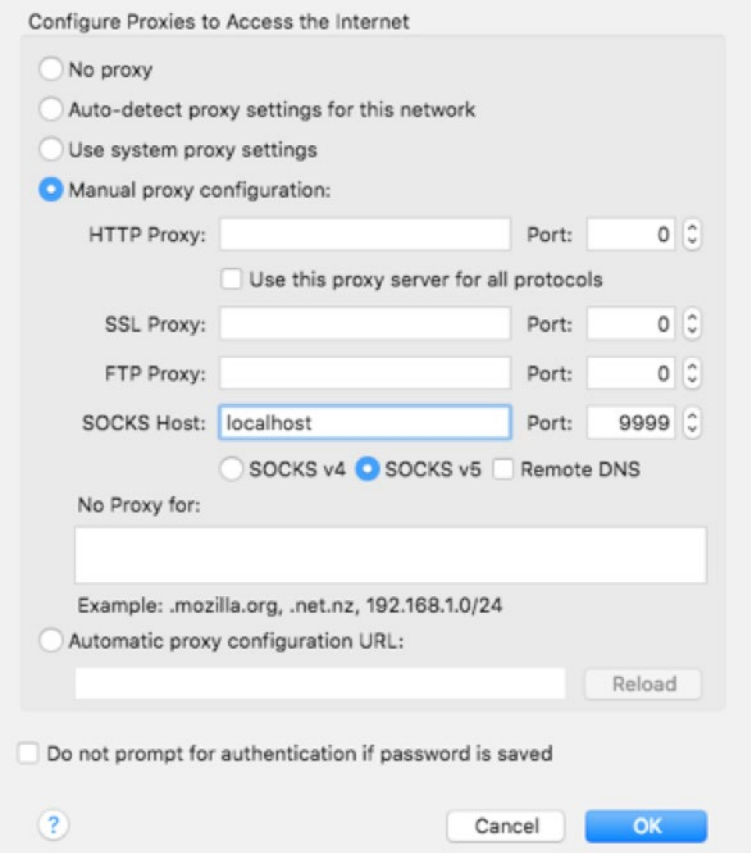
In order to overcome the /sbin/nologin restriction mentioned above, an attacker might leverage another interesting SSH option, **-N**, which the SSH's man page describes as follows:

> *"Do not execute a remote command. This is useful for just forwarding ports (protocol version 2 only)."*

The combination of the -D and -N SSH options allows a remote user, who knows the The combination of the -D and -N SSH options allows a remote user, who knows the credentials for the "admin" user of the device (which have publicly known factory default values), to turn the device into a SOCKS proxy and to leverage it for malicious activity – despite the fact that the configuration on the device was supposedly "hardened". Similarly, the attacker can use the -L and -N options to configure explicit proxying for use with attack tools that are not easily adapted to using SOCKS.

A quick verification on our lab device showed that this weakness was indeed exploitable quite easily.

First, an attacker has to form the SSH tunnel. At this point, the SSH tunnel is used like any other kind of SOCKS proxy, and all the user has to do is configure the relevant network process (e.g. Browser, script, attack tool, etc.) to use that (local) SOCKS proxy. For example:



Any traffic routed through this SSH tunnel will appear as if it is originating from the NVR device's IP address.

To the root causes that allow for this vulnerability are:

- The device came with a factory-default administration account whose credentials are publicly known (e.g. admin:admin)
- The device allowed remote SSH connections
- The SSH daemon was configured to allow TCP Forwarding (AllowTcpForwarding = true), which rendered the /sbin/nologin hardening technique useless for this type of attack

From NVR to IoT / We now understand that remote users are harnessing the NVR device's weak factory default configuration for launching account checking attacks. With that knowledge, we decided to check if there are other types of IoT devices that carry the same weakness and were being abused for similar account checking purposes. We found a very wide range of devices that are actively being exploited, such as:

- CCTV, NVR, DVR devices (video surveillance)
- Satellite antenna equipment
- Networking devices (e.g. Routers, Hotspots, WiMax, Cable and ADSL modems, etc.)
- Internet connected NAS devices (Network Attached Storage)

In some of the devices, we managed to find additional weak factory default configurations, which can be used to avoid the need to provide default credentials altogether, such as:

- A very common router brand that has a "root" **privileged** user, with a factory default password, but without the ability to login (similar to the NVR device mentioned earlier)
- A common wireless hotspot, which doesn't require any password for SSH connections

At least one network-device vendor published a ***security advisory*** disclosing the potential danger related to unauthenticated TCP tunneling through the SSH daemon installed on its own devices. This was one of the affected device types discovered during our research.

From IoT to Internal Network Compromise / Once an IoT device allows a remote user to form an SSH tunnel, and to use it as a SOCKS proxy, the attacker is not limited to only mounting attacks against **Internet-facing** servers, but also as a "beachhead" to launch attacks against the **internal** network hosting the Internet-connected device.

We managed to confirm and validate the feasibility of this severe abuse-case in our lab environment, and believe that malicious users are and will continue to actively exploiting this to penetrate private networks.

We are actively looking at mitigation strategies, and will continue to update our customers on how the best protect their customers from vulnerabilities like this one.

About the Authors

Ezra Caltum / Sr. Security Research, Team Leader, Akamai

Ory Segal / Sr. Director of Threat Research, Akamai

Both authors are a part of Akamai's Threat Research team, which is responsible for researching, designing and developing the security logic powering Akamai's KONA line of security products.

Additional Acknowledgements

Several additional Akamai Threat Research team members contributed data analysis, which led to the discovery described in this paper. We would like to acknowledge their contribution:

Aharon Fridman / for researching and developing Akamai's account checking behavioral heuristics, which are responsible for the detection and tracking of Account Takeover campaigns

Or Katz, Adir Gozlan / for providing deep research on trends and statistics in Account Takeover campaigns and exposing the mass scale involved

Ryan Barnett / for his initial attention to the IoT nature of these attacks and his original blog post on this topic

SIRT Team / for their research assistance and support in digging into the details of this

About Akamai® As the global leader in Content Delivery Network (CDN) services, Akamai makes the Internet fast, reliable and secure for its customers. The company's advanced web performance, mobile performance, cloud security and media delivery solutions are revolutionizing how businesses optimize consumer, enterprise and entertainment experiences for any device, anywhere. To learn how Akamai solutions and its team of Internet experts are helping businesses move faster forward, please visit *www.akamai.com* or *blogs.akamai.com*, and follow *@Akamai* on Twitter.

Akamai is headquartered in Cambridge, Massachusetts in the United States with operations in more than 57 offices around the world. Our services and renowned customer care are designed to enable businesses to provide an unparalleled Internet experience for their customers worldwide. Addresses, phone numbers and contact information for all locations are listed on *www.akamai.com/locations*.
