Honeynet Project's FC6 "Analyzing Malicious Portable Destructive File": my submission

publié par malphx le dimanche, décembre 26 2010 - 15:23

The Honeynet Project's team has published the results for the 6th Forensic Challenge 2010.

My official results:

{{For this 6th challenge, we received a total of 21 submissions. With your score of 20, you came into position 7.

Below you will find your score per answer:

```
 Answer 1: 1 points Answer 2: 2 points Answer 3: 3 points Answer 4:
1 points Answer 5: 0.5 points Answer 6: 1 points Answer 7: 2.5
points Answer 8: 4 points Answer 9: 2 points Answer 10: 1 points
Answer Bonus 1: 1 points Answer Bonus 2: 1 points
```

This was a competitive challenge. The top three submissions were within a point of a total score and many submissions that didnt place in the top three were close. We hope you enjoyed the challenge and learned a bit.

The top three submissions have been posted to the challenge web page at http://honeynet.org/challenges/2010_6_malicious_pdf and we encourage you read through them.

We will be taking a little break, but will continue our challenges in 2011. We hope to see your submissions!}}

Well, this time the competition was really hard, as you can see with a score of 20/22 I came in position 7.Others were better ! :-)

If you're interrested my submission for FC6 is in attachment.

Feel free to leave a comment !

Playing with SIP, NMAP and NSE, now writing a SIP library...

publié par malphx le lundi, août 23 2010 - 23:54

Since my last post, I finally decided to start writing a SIP library for nmap.
This lib will be minimalist and be largely based on the http.lua library taken from Nmap 5.0

It will be used by two NSE scripts:

- sip-extscan.nse: a script which try to list (find) valid SIP extensions on a SIP registrar
- sip-brute.nse: a script that try to bruteforce SIP extensions password on a registrar

The target used for the test is a Tribox based host (Asterisk PBX 1.6.0.26-FONCORE-r78)
With actually four extensions:

- Actually, it will try to register with the extension name against the password and use the unpwdb library

```
sudo nmap -sU -p U:5060 -T5 --script
sip-map2,sip-extscan3,sip-brute2 --script-args
exten_range="5000-5010" 172.17.0.53

Starting Nmap 5.00 ( http://nmap.org ) at 2010-08-23 23:20 CEST
Interesting ports on 172.17.0.53:
PORT STATE SERVICE
5060/udp open sip
|_ sip-map2: SIP 2.0 device detected
| sip-extscan3:
| Unprotected Extensions
| 5003
| Protected Extensions
| 5000
| 5001
|_ 5002
| sip-brute2:
| exten: 5001 Password: 1234
|_ exten: 5002 Password: 1234

Nmap done: 1 IP address (1 host up) scanned in 107.24 seconds
```

It seems that the work is in the good way, however, a lot of testing must still be done.

Playing with SIP, NMAP and NSE

publié par malphx le mardi, août 17 2010 - 19:04

In the last Honeynet Project's Forensic Challenge (FC4), one question (Section 1, question 2) caught my attention.
It was about the possibility that the given log file could have been generated using a simple Nmap UDP scan.
In the challenge, the answer was : No.
Because a simple Nmap's UDP scan uses UDP packets without any payload and thus could not generate valid SIP requests.
But, Nmap offers a powerful scripting engine: Nmap Scripting Engine or NSE.

With NSE it is possible to interact with the targetted host using simple to complex communication exchanges.

After having read the NSE part of the Nmap book, I decided to give a try at NSE.
My first NSE script (modestly) behaves like the SIPvicious tool: svmap.py.

This script, named sip-map.nse tries to find valid SIP server by sending a SIP OPTIONS request using the UDP protocol.

Usage:

```
# Without version (User-Agent) information
sudo nmap -sU -p U:5060 –script sip-map.nse
# With version information
sudo nmap -sU -p U:5060 -sV –script sip-map.nse
```

Output:

```
Interesting ports on X.X.X.X:
PORT STATE SERVICE VERSION
5060/udp open sip Asterisk PBX 1.6.0.26-FONCORE-r78
|_ sip-map: SIP 2.0 compliant device detected
```

sip-map.nse is the first script from a series of scripts I wish to write.
These scripts will be about SIP scanning with a behaviour close to the SIPvicious tools but using Nmap.

You can download it here: sip-map.nse

Feel free to leave a comment !

Honeynet Project's FC4 "VoIP": my submission

publié par malphx le dimanche, juillet 25 2010 - 10:50

The Honeynet Project's team has published the results for the 4th Forensic Challenge 2010 VoIP.

My official results:

> Thank you for participating in the 4th Honeynet Project Forensic Challenge 2010: VoIP.
> Sjur, Ben, Jianwei, Roland, and Julia finished evaluating your submission. You have received a total of 62 of 63 points.
> Below you will find your score per answer:
>
> - *Answer 2: 6 points*
>
> A sample solution as well as the submissions of the winners has been posted to the challenge web page at http://honeynet.org/challenges/2010_4_voip. Sjur, Ben, Jianwei, Roland, and Julia will be summarizing highlights from various submissions in a blog post shortly.
>
> We are still finalizing our next challenge. Please subscribe to our RSS feed or check our web sites for announcements.
>
> For this 4th challenge, we received a total of 21 submissions. With your score of 62, you came into position 1. Congratulations!!!!

You could find my submission for FC4 on the Honeynet Project's site.For this one, I've used a great visualization tool named PicViz written by Sébastien Tricaud from the French Chapter.
You should read his paper about his tool: Know Your Tools: use Picviz to find attacks

Feel free to leave a comment !

My submission to the Honeynet Project Forensic challenge 2010/3

publié par malphx le vendredi, mai 14 2010 - 22:23

The Honeynet Project's team has published the results for their last forensic challenge.

Congratulations to the Winners !

This time, luck was not with me and I was not in the top 3, but came only in 4th position.

Because only the winners submissions are published on the Challenges official site, I publish mine here for review and comments.

Unfortunately, I misunderstood question 5…

This challenge was really good and again taught me a lot of new things and new tools.

My official results:

For this 3rd challenge, we received a total of 22 submissions. With your score of 41, you came into position 4. You placed into the top third. With the many great submissions and the competitive field, this is a great accomplishment. Congratulations.

Below you will find your score per answer:
- Answer 1: 2 points
- Answer 2: 4 points
- Answer 3: 2 points
- Answer 4: 4 points
- Answer 5: 1 points
- Answer 6: 8 points
- Answer 7: 2 points
- Answer 8: 6 points
- Answer 9: 4 points
- Answer 10: 5 points

In addition, you have received 3 bonus points.

You could find my submission here:
franck_dot_guenichot_at_orange_dot_fr_Forensic_Challenge_2010_-_Challenge_3.pdf

Feel free to leave a comment !

Honeynet Project's FC2010/2 - My submission

publié par malphx le mercredi, mars 24 2010 - 13:48

The second 2010´char(180) Forensic Challenge published by the Honeynet Project is now closed, and the results have been published.
This time the investigators (or contestants) had to dissect a pcap file containing network traces of browsers under attack.
The analysis revealed that a lab setup has been used to mimic the interactions between victim's browsers and some malicious Web sites.
Feel free to review my submission, all the winners submissions and the solution given by the Honeynet Project's Team.
I haved scored 43/43 for this one, and so I'm one of the 4 winners
I'm now waiting the publication of #3 ('Banking Troubles'), which promises to be very interresting.
Finally, I would also like to thank all the Honeynet Project's team for giving us such

interresting and educationnal contests !

Honeynet: Challenge 1 of the Forensic Challenge 2010

publié par malphx le mardi, février 16 2010 - 13:49

I've participated in the last Honeynet Challenge.
This Challenge ran from Jan 18th 2010 to Feb 15th 2010.

It was about the analysis of a PCAP trace file containing an attack.
Results have been published,
and I'm proud of my #2 position in this contest.
You could find my submission on The Honeynet Project's website.

Another solution to the Network Forensics Puzzle #3

publié par malphx le samedi, février 6 2010 - 11:13

For this contest, We've had to deal with Apple TV traffic.

Tools Used:
_ TShark 1.2.2
_ macfinder.rb (custom ruby script giving IP/MAC bindings from a pcap file.) (require packetfu)
_ httpdumper (custom ruby script that can display and dump HTTP conversations) (require packetfu and terminal-table)
_ plist.rb (custom ruby script that can display informations extracted from Apple Property-List 1.0 XML documents)
Recommandations for using these tools:
macfinder and httpdumper rely on the wonderful ruby lib: packetfu.
Significant performances improvements have been done by its author in version 0.3.1. (8x faster)
So, You **SHOULD use the last version of packetfu**, eg: at least, packetfu 0.3.1
.

Detailled Answers As usual, we have to verify the evidence file integrity:

```
franck@ODIN:~/Analysis/Sources/Puzzle_3$ md5sum evidence03.pcap

f8a01fbe84ef960d7cbd793e0c52a6c9 evidence03.pcap
```

Ok, we're good to go !

First,I used to look at the protocol hierarchy stats given by tshark to take a first look at a pcap file:

```
franck@ODIN:~/Analysis/Sources/Puzzle_3$ tshark -r evidence03.pcap
-qz io,phs
===================================================================
Protocol Hierarchy Statistics
Filter: frame

frame frames:1778 bytes:1508750
 eth frames:1778 bytes:1508750
 ip frames:1778 bytes:1508750
 udp frames:28 bytes:6102
 dns frames:28 bytes:6102
 tcp frames:1750 bytes:1502648
 http frames:167 bytes:93189
 image-gif frames:33 bytes:21202
 xml frames:18 bytes:20852
 tcp.segments frames:65 bytes:46469
 http frames:65 bytes:46469
 xml frames:17 bytes:11732
 image-jfif frames:48 bytes:34737
===================================================================
```

Hum, I bet we'll have to work with HTTP and some XML documents/data !

Let's continue…

We know that Ann has recently aquired an AppleTV and has configured it with a static IP address: 192.168.1.10
For some obvious reason, we have to know the mac (or hardware) address of Ann's new HDTV box.
Tshark could help for this task:

```
franck@ODIN:~/Analysis/Sources/Puzzle_3$ tshark -r evidence03.pcap
-R "ip.src==192.168.1.10" -Tfields -e "eth.src" |uniq
00:25:00:fe:07:c4
```

I've also coded a small ruby script for this task: macfinder.rb. Here's the help screen:

```
franck@ODIN:~/Analysis/Sources/Puzzle_3$ ./macfinder.rb

 macfinder version 0.1
```

```
 Copyright © 2009 Franck GUENICHOT
 macfinder comes with ABSOLUTELY NO WARRANTY;
 This is free software, and you are welcome
 to redistribute it under certain conditions.
 (GPL v3)

 Usage: macfinder [options]
 -i, –ip  Display Mac address for the given IP address only
(4-digit decimal dot notation form)
 -v, –version Display version information
 -h, –help Display this screen
```

Without any switch, macfinder.rb displays all the source IP/MAC address found in the pcap file:

```
franck@ODIN:~/Analysis/Sources/Puzzle_3$ ./macfinder.rb
evidence03.pcap
Listing all Mac Address found !
IP: 8.18.65.10 | Mac: 00:23:69:ad:57:7b
IP: 8.18.65.32 | Mac: 00:23:69:ad:57:7b
IP: 8.18.65.88 | Mac: 00:23:69:ad:57:7b
IP: 8.18.65.89 | Mac: 00:23:69:ad:57:7b
IP: 8.18.65.67 | Mac: 00:23:69:ad:57:7b
IP: 8.18.65.58 | Mac: 00:23:69:ad:57:7b
IP: 8.18.65.82 | Mac: 00:23:69:ad:57:7b
IP: 8.18.65.27 | Mac: 00:23:69:ad:57:7b
IP: 192.168.1.10 | Mac: 00:25:00:fe:07:c4
IP: 4.2.2.1 | Mac: 00:23:69:ad:57:7b
IP: 66.235.132.121 | Mac: 00:23:69:ad:57:7b
```

From the listing above we can easily find Ann’s AppleTV mac address.
But to be less verbose, and because we know the IP address, we can use the -i switch to display only the interresting MAC.
Here’s the help screen:

```
franck@ODIN:~/Analysis/Sources/Puzzle_3$ ./macfinder.rb -i
192.168.1.10 evidence03.pcap
Mac: 00:25:00:fe:07:c4
```

And Voila !
Going Deeper Part I : HTTP

Now, we have to go deeper in the pcap file to analyse Ann's networking activity and particularly
her AppleTV network conversations.
Tshark let us know that we'll have to deal with HTTP (and maybe XML documents, later), so I wrote a specialized tools
to facilitate the investigation: httpdumper
httpdumper basically displays informations about HTTP conversations. The HTTP protocol is a Request/Response protocol meaning
that a client makes a request to a server with HTTP request messages and the server answers with HTTP response messages.
httpdumper handles this mechanism and displays these conversations in an easy to understand manner.
Some terminology:
An HTTP conversation, for httpdumper, is the set of all REQUEST/RESPONSE HTTP messages involving the same 2 hosts and tcp ports.
An HTTP flow is an unidirectionnal flow of http data (eg: client to server (request) or server to client (response)

Here's the help screen:

```
franck@ODIN:~/Analysis/Sources/Puzzle_3$ ./httpdumper -h

 httpdumper version 0.1
 Copyright © 2010 Franck GUENICHOT
 httpdumper comes with ABSOLUTELY NO WARRANTY;
 This is free software, and you are welcome
 to redistribute it under certain conditions.
 (GPL v3)

 Usage: httpdumper [options] -r
 -r, –read  Read the given pcap file [REQUIRED]
 -c, –conversation # List only flows for conversation #
 -f, –flow # List only flow #
 –with-headers For Display ONLY
 -d, –dump Dump the selected conversation or flow
 -p, –port  Define custom HTTP port
 -s, –stats type,[val1],[val2] Displays statistics
 Valid options:
 Request stats: request,[requester_ip],[requested_host]
 URI list: uri,[requester_ip],[target_hostname]
 -v, –version Display version information
 -h, –help Display this screen
```

httpdumper is only a passive (lightweight) analysis tool, it needs a file in entry, so -r options is required to launch this tool.
The default output (without any options) displays all the HTTP conversations found in the given pcap file.

Let's do it !

```
franck@ODIN:~/Analysis/Sources/Puzzle_3$ ./httpdumper -r
evidence03.pcap
Reading file evidence03.pcap
Parsing packets…
1778 packets read in 4.385 sec.

Found 20 HTTP conversation(s)
+———————+————————————+———————+———+———+———————+
| Conversation Index | Hosts | HTTP Flow count | Request | Response
| Cumulative length |
+———————+————————————+———————+———+———+———————+
| 0 | 192.168.1.10:49163  8.18.65.67:80 | 2 | 1 | 1 | 16551 |
| 1 | 192.168.1.10:49164  66.235.132.121:80 | 2 | 1 | 1 | 43 |
| 2 | 192.168.1.10:49165  8.18.65.32:80 | 8 | 4 | 4 | 22453 |
| 3 | 192.168.1.10:49166  66.235.132.121:80 | 8 | 4 | 4 | 172 |
| 4 | 192.168.1.10:49167  8.18.65.58:80 | 20 | 10 | 10 | 161118 |
| 5 | 192.168.1.10:49168  8.18.65.67:80 | 4 | 2 | 2 | 3157 |
| 6 | 192.168.1.10:49169  66.235.132.121:80 | 2 | 1 | 1 | 43 |
| 7 | 192.168.1.10:49170  8.18.65.82:80 | 44 | 22 | 22 | 675124 |
| 8 | 192.168.1.10:49171  8.18.65.27:80 | 6 | 3 | 3 | 13582 |
| 9 | 192.168.1.10:49172  66.235.132.121:80 | 6 | 3 | 3 | 129 |
| 10 | 192.168.1.10:49173  8.18.65.27:80 | 8 | 4 | 4 | 12744 |
| 11 | 192.168.1.10:49174  66.235.132.121:80 | 2 | 1 | 1 | 43 |
| 12 | 192.168.1.10:49175  66.235.132.121:80 | 6 | 3 | 3 | 129 |
| 13 | 192.168.1.10:49176  8.18.65.67:80 | 4 | 2 | 2 | 3493 |
| 14 | 192.168.1.10:49177  8.18.65.10:80 | 32 | 16 | 16 | 362826 |
| 15 | 192.168.1.10:49178  66.235.132.121:80 | 2 | 1 | 1 | 43 |
| 16 | 192.168.1.10:49179  8.18.65.88:80 | 20 | 10 | 10 | 5576 |
| 17 | 192.168.1.10:49180  66.235.132.121:80 | 20 | 10 | 10 | 430 |
| 18 | 192.168.1.10:49181  8.18.65.89:80 | 18 | 9 | 9 | 4861 |
| 19 | 192.168.1.10:49182  66.235.132.121:80 | 18 | 9 | 9 | 387 |
+———————+————————————+———————+———+———+———————+
```

The table above show all the HTTP conversations found. (this kind of table is best viewed on large display)
The flow count indicates the number of flows in each conversations
Request and Response column, each displays the number of HTTP Request or HTTP response in each conversation
Cumulative length: the length (in Bytes) of the HTTP Payloads (or HTTP message body) exchanged in each conversation.
Note: this length takes only HTTP Response payloads into account. By now HTTP Request message body, if any, is not displayed (and not dumpable")

Quickly, we learn interresting infos:
_ 20 HTTP conversations, all involving the same client (Ann's AppleTV)
_ 7 of them are composed of 18+ flows
_ Conversation #7 has the greater cumulative length (so, the largest HTTP payload)

Network forensics contest Puzzle#2: my solution

publié par malphx le mardi, novembre 24 2009 - 23:32

But all these informations aren't enough: to continue our investigation we need to go deeper.
Let's try to answers Question #2: What User-Agent string did Ann**Update:** Well, results have been published, and (Wow !) I'm one of the 2 winners of this challenge. What a great surprise ! A lot of good work have been done by the other finalists, too. You really have to view their submissions.

Now that the deadline is past, and the official answers have been published on the Network Forensics Puzzle Contest.
it's now time for me to publish my own submission.
For this one, i've written 2 tools in ruby.The first is named smtpdump and could be used to retrieve interresting informations on SMTP conversations in a pcap file. The second docxtract is able to extract files from a docx archive.

Well, this time, it seems the challenge will be hard…
Some of the contestants have already published their own solutions or tools, and all the solutions i've already read so far are really good ones !

Adéquation musicale

publié par malphx le dimanche, novembre 25 2007 - 23:48

…moi.

Je ne sais pas si c'est vraiment interressant, mais ce ne serait pas la première chose inutile publiée sur le WEB ;)
Si nous sommes en pleine harmonie musicale ou pas, faites me le savoir…

Crackme#4: le dernier pour l'instant

publié par malphx le dimanche, novembre 25 2007 - 17:57

Toujours écrit en ASM et packé avec mon protector perso.
D'ailleurs c'est la partie qui m'interresse le plus dans cette discipline, je parle de la partie protection de PE
Car vous le verrez, une fois unpacké, le crackme en lui-même est facile.
Publication et solution sur crackmes.de

Le readme:

```
It's time for crackme4.
I wish that you will have fun defeating it ! If U can :p
=============================================================
Code: ASM
Protection: Custom
Level: 3 (i think)
=============================================================
Objectives:
Required: Make a keygen or patch (but a keygen is better)
Required: submit a tuto on crackmes.de
Required: Rate it !
Optional: Unpack
Optional: Describe the protector.
=============================================================
Rules:
>>> everything is allowed but,
>>> serial fishing is not a valid solution
=============================================================
tested on WinXP PRO SP2, WinXP Home SP1, and Win2K3 server
Entreprise Edition.
You could send me your solution or bug report (email is in my
crackmes.de profile)
```

```
Good work !
Znycuk
```

[znycuk crackme#4]

Pour l'instant c'est le dernier de la série, j'espère bien continuer et sortir #5 pour lequel j'avais quelques idées, cependant le temps me fait défaut.
Plus tard peut-être…

Crackme#3

publié par malphx le dimanche, novembre 25 2007 - 17:46

Le 3eme de la série, celui ci est aussi protégé par un packer custom
[Diffusé en juin 2006 sur crackmes.de]
Il a évidemment été solutionné par les reversers du site mentionné plus haut.
Voici le readme original:

```
Hi,

Here is cracKme#3, i think this one will be hard for real newbies.
But not for the good reversers from crackmes.de :-)

I wish that you will have fun defeating it !

Note: My RDG Packer Detector seems to have a false positive on it.



================================================================
Code: ASM
Protection: Custom
Level: 2 (maybe)
================================================================

Objectives:

_ Unpack if needed
_ Make a keygen
_ submit a tuto (it should also describe the protector)



================================================================
```

```
Rules:
>>> Patching the goodboy jump is not allowed
 As a Keygen is required: no serial fishing
```

Crackme#2 "Find my passw0rd"

publié par malphx le dimanche, novembre 25 2007 - 17:37

Celui ci aussi a bien évidemment été publié sur crackmes.de, en mai 2006.

Le readme:

```
Hi,
This is my crackme # 2 Find my passw0rd.
In this one, i've try to make the job a little harder.

Code: ASM
Protection: Custom
Level: 2

Objectives:
_ Unpack if needed
_ Find the password
_ submit a tuto

Rules:
Patching not allowed…

tested on WinXP SP2 but should work on SP1 and Win2K.

Good work !
Znycuk
```

znycuk crackme2 Find my Passw0rd
Pour celui là, j'avais commencé un projet de packer/protector très simple.

Crackme#1

publié par malphx le dimanche, novembre 25 2007 - 17:13

Le fichier readme original, posté sur le site crackmes.de

```
Hi all,
This is my first Keygenme, written in ASM.
_ Find the solution
_ make a keygen
_ And submit a tuto
tested on WinXP SP2 but should work on SP1 and Win2K.
Good work !
Znycuk
```

Ah oui, je poste sous le pseudo znycuk sur le site crackmes.de,

je vous laisse trouver le lien avec mon pseudo original…

znycuk keygenme#1

La solution pour ce crackme peut être trouvée sur le site crackmes.de.

a+

Modification d'un firmware pour le DG834G --Troisieme partie--

publié par malphx le dimanche, novembre 11 2007 - 23:17

ATTENTION
Cette manipulation est de loin la plus dangereuse pour votre routeur.
Si vous n'avez jamais customisé/recompilé de noyau linux,
alors je vous déconseille fortement de vous lancer directement sur la recompilation du kernel pour le DG834G.
Pour les autres, dans la plupart des cas, la modification du kernel n'est pas nécessaire.
Elle n'est nécessaire que pour l'ajout d'une fonctionnalité nécessitant un support au niveau du noyau.

Préparation des sources

Renommage du repertoire sourceLors de la compilation du noyau le nom d'origine (**DG834(G).V2.10.17_src**) pose problème à cause des parenthèses.

1. On peut le renommer par exemple en V2.10.17_src, ou tout autre nom ne comportant pas de caractères spéciaux.`mv DG834(G).V2.10.17_src V2.10.17_src`

Application du patch patch-knlDepuis le repertoire source du firmware (maintenant V2.10.17) , les sources du noyau se trouvant dans le repertoire **linux-2.4.17_mvl21**

```
[root@evectra V2.10.17_src]# lsapps DG834B_V2.10.17.img
linux-2.4.17_mvl21patch-apps README target.tar.bz2 uClibc-0.9.19
build.sh fs.bin MonFirmwareCustom.img patch-knl target tools
[root@evectra V2.10.17_src]# patch -p0 sortie
tronquée******************patching file
linux-2.4.17_mvl21/net/Makefile.ipsecmd5patching file
linux-2.4.17_mvl21/net/Makefile.preipsecpatching file
linux-2.4.17_mvl21/net/Makefile.wipsecpatching file
linux-2.4.17_mvl21/net/atm/Makefilepatching file
linux-2.4.17_mvl21/net/atm/br2684.cpatching file
linux-2.4.17_mvl21/net/atm/common.cpatching file
linux-2.4.17_mvl21/net/atm/pppoatm.cpatching file
linux-2.4.17_mvl21/net/atm/pppoatm.c.3.5patching file
linux-2.4.17_mvl21/net/atm/pvc.cpatching file
linux-2.4.17_mvl21/net/ax25/ax25_ds_in.cpatching file
linux-2.4.17_mvl21/net/ax25/ax25_ds_subr.cpatching file
linux-2.4.17_mvl21/net/bluetooth/af_bluetooth.cpatching file
linux-2.4.17_mvl21/net/bluetooth/hci_core.c***********Le patch modifie
```

les sources du noyau original de montavista, il ajoute le support des éléments spécifiques à la plateforme du routeur.
Point important, le patch créé aussi le fichier **.config** qui contient toute la configuration pour la compilation par défaut.
En clair, on peut dire qu'après l'application du patch, le noyau serait prêt à être compilé.

Modification/personalisation

Que vous dire ici, sinon que cela se passe exactement comme pour un noyau linux :D.
Mais à moins de savoir exactement ce que vous faites, je vous déconseille de modifier le paramètrage d'origine… (retrait d'option / modification de valeur pour la flash…)

Que faire alors ?A vous de voir pourquoi vous voulez recompiler ce noyau ;)

Quelques idées:Ajout de fonction au niveau ipsec, ajout de modules iptables pour profiter de toute la puissance de ce parefeu, etc…

Compilation

Vos modifications sont faites ?
il ne reste plus qu'à compiler et générer une image intégrable dans le firmware.
Pour cette partie, vous aurez besoin de l'utilitaire dgfirmware qui va permettre de fusionner le noyau avec l'image du firmware.
De plus, j'ai utilisé une toolchain différente: mips-fp-le, issue du previewkit de montavista.
Malheureusement, je n'ai plus le lien :(, mais comptez sur moi pour le mettre en ligne dès que possible.

Les étapes de la compilation:

- make menuconfig
- make dep
- make ram_zimage_pad

Voila, on peut trouver maintenant dans le repertoire des sources du noyau un fichier nommé **ram_zimage_pad.bin**.
C'est ce fichier que nous allons fusionner dans l'image du firmware, et ce, à l'aide de **dgfirmware**
Voici la syntaxe de cet utilitaire:

```
[root@evectra DG834G.V2.10.09_src]# ./dgfirmware -husage:
dgfirmware [|opts|] |img| |img| firmware image filename |opts|
-h print this message -f fix the checksum -x |file| extract the
rootfs file to |file| -xk |file| extract the kernel to |file|
-m |file| merge in rootfs filrom |file| -k |file| merge in
kernel from |file| -w |file| write back the modified firmware
```

La commande sera donc de la forme: **./dgfirmware -f -k ram_zimage_pad.bin -w MonfirmwareCustom.img DG834G_V2.10.17.img**
Ou:

-f => fixe le checksum de la nouvelle image (calcul du CRC)
-k ram_zimage_pad.bin => image du noyau à fusionner
-w MonfirmwareCustom.img => Nom de la nouvelle image intégrant notre kernel
DG834G_V2.10.17.img => Image du firmware original.

```
[root@evectra DG834G.V2.10.09_src]# ./dgfirmware -f -k
ram_zimage_pad.bin -w MonfirmwareCustom.img DG834G_V2.10.17.img
Read firmware file
Firmware product: DG834
Firmware version: 1.42.09
Read kernel file
image checksum = 1f86
real checksum = ab3c
Bad Checksum, fix it
Write image file
```

Bien que la sortie indique un **Bad Checksum**, dgfirmware a bien corrigé le checksum.Pour en être sur:

```
[root@evectra DG834G.V2.10.09_src]# ./dgfirmware -f
MonfirmwareCustom.img
Read firmware file
Firmware product: DG834
```

```
    Firmware version: 1.42.09
image checksum = ab3c
real checksum = ab3c
Checksum is correct, good
```

Ben on dirait que ça y est :D
Il ne reste plus qu'à tester en flashant le routeur.

Mais n'oublie pas que l'utilitaire de recovery est ton ami :)

Modification d"un firmware pour le DG834G --Deuxieme partie--

publié par malphx le dimanche, novembre 11 2007 - 23:04

Pour plus d'info sur le Wake on Lan, voici un mini-howto.

On y est ! Notre machine Linux est prête, on peut commencer la modif.

Inspection rapide du filesystem 'target'le voici:

```
[root@evectra target]# ls -l
total 44
drwxr-xr-x 2 root root 4096 jui 2 2004 bin
drwxr-xr-x 2 root root 4096 mai 20 2001 dev
lrwxrwxrwx 1 root root 8 mai 25 23:45 etc -> /tmp/etc
drwxr-xr-x 3 root root 4096 jui 2 2004 lib
drwxr-xr-x 2 root root 4096 nov 13 2000 proc
drwxr-xr-x 2 root root 4096 jui 2 2004 sbin
drwxr-xr-x 2 root root 4096 jui 28 2000 tmp
drwxr-xr-x 8 root root 4096 mar 12 2004 usr
lrwxrwxrwx 1 root root 8 mai 25 23:45 var -> /tmp/var
lrwxrwxrwx 1 root root 8 mai 25 23:45 www -> /tmp/www
drwxr-xr-x 2 root root 4096 jun 3 2004 www.deu
drwxr-xr-x 2 root root 4096 jui 1 2004 www.eng
drwxr-xr-x 2 root root 4096 mai 13 2004 www.fre
drwxr-xr-x 2 root root 4096 jun 3 2004 www.ita
```

On retrouve la structure familière de ce bon vieux pingouin…
A noter tout de même, les liens symboliques **etc**,**var** et **www**.
Comme indiqué dans un précédent article: après le boot du DG, seul **/tmp** sera en lecture/ecriture puisque situé en RAM.
Le reste des repertoires seront en lecture seule.

Compilation de notre utilitaire

L'utilitaire sera donc **ether-wake v1.09** de Donald Becker (ether-wake.c: v1.09 11/12/2003 Donald Becker,http://www.scyld.com/).
Les sources sont disponibles ici.
Note: Les sources ont été légèrement modifiées pour pouvoir être linké avec ucLibc. Enf effet, le programme original faisait appel à des fonctions non implémentées dans la librairie C du DG834(G).
Les fonctions en question ont simplement été commentées. L'utilitaire final dans son utilisation sur le DG834(G) ne souffre pas de ces modifications.

La compilation proprement dite se passe très simplement:

```
[root@evectra WOL]# mipsel-uclinux-gcc -Wall -o ether-wake
ether-wake.c
[root@evectra WOL]# ls -l
total 36
-rwxr-xr-x 1 root root 21283 mai 26 13:22 ether-wake
-rw-r–r– 1 franck franck 11113 mar 29 22:00 ether-wake.c
[root@evectra WOL]#
```

Après avoir saisi la commande **mipsel-uclinux-gcc -Wall -o ether-wake ether-wake.c**, on obtient un executable MIPS:

```
[root@evectra WOL]# file ether-wake
ether-wake: ELF 32-bit LSB MIPS-I executable, MIPS, version 1
(SYSV), dynamically linked (uses shared libs), not stripped
```

On remarquera qu'il est linké dynamiquement à ucLibc.

```
[root@evectra WOL]# mipsel-uclinux-ldd ether-wake
libc.so.0 => /opt/brcm/hndtools-mipsel-uclibc-0.9.19/lib/libc.so.0
(0x00000000)
/lib/ld-uClibc.so.0 => /lib/ld-uClibc.so.0 (0x00000000)
```

Il ne nous reste plus qu'a stripper notre binaire pour gagner quelques octets:

```
[root@evectra WOL]# mipsel-uclinux-strip ether-wake
[root@evectra WOL]# ls -l ether-wake
-rwxr-xr-x 1 root root 15048 mai 26 13:26 ether-wake
```

Et voila, notre utilitaire est prêt à l'intégration dans le firmware.

Intégration au firmware

Le plus simplement du monde, on copie le binaire ether-wake dans un repertoire de notre filesystem ‘target’.
/usr/sbin par exemple.
Notre filesystem target est modifié, il ne nous reste plus qu’a construire l’image de notre firmware custom.

Reconstruction de l’image du firmware

Netgear, nous fournit tous les outils permettant cette reconstruction.
Voici comment les utiliser pour construire notre image.
Depuis la racine du repertoire créé lors de l’extraction des sources du firmware:

```
[root@evectra V2.10.17_src]# lsapps build.sh DG834B_V2.10.17.img
linux-2.4.17_mvl21 patch-apps patch-knl README target
target.tar.bz2 tools uClibc-0.9.19
```

On va utiliser le script **build.sh**, dont la syntaxe est la suivante:
build.sh [Nom du firmware source] |[Nom du FS target] [Nom du firmware destination]
Dans notre cas:

```
[root@evectra V2.10.17_src]# ./build.sh DG834B_V2.10.17.img target
MonFirmwareCustom.imgOriginal
Image: DG834B_V2.10.17.img
Your Filesystem: target
New Image: MonFirmwareCustom.img
Press ‘y’ to continue
```

On presse ‘y’, c’est parti:
Note: suivant la puissance de votre machine, cette étape peut être un peu longue.

```
Creating little endian filesystem on fs.bin,
block size 32768.
Little endian filesystem,
data block size 32768,
compressed data,
compressed metadata,
compressed fragments
Filesystem size 1697.66 Kbytes (1.66 Mbytes)
26.56% of uncompressed filesystem size (6391.35 Kbytes)
Inode table size 3147 bytes (3.07 Kbytes)
35.03% of uncompressed inode table size (8985 bytes)
Directory table size 2651 bytes (2.59 Kbytes)
55.19% of uncompressed directory table size (4803 bytes)
```

```
Number of duplicate files found 1Number of inodes 393
Number of files 267
Number of fragments 39
Number of symbolic links 67
Number of device nodes 32
Number of fifo nodes 0
Number of socket nodes 0
Number of directories 27
Number of uids 1 unknown (0)
Number of gids 0
MonFirmwareCustom.img Created!
```

Et voila, notre nouveau firmware est créé dans le repertoire courant

```
[root@evectra V2.10.17_src]# lsapps DG834B_V2.10.17.img
linux-2.4.17_mvl21 patch-apps README target.tar.bz2
uClibc-0.9.19build.sh fs.bin MonFirmwareCustom.img patch-knl target
tools
```

Il ne reste plus qu'a l'uploader sur notre routeur par la procédure habituelle. Ensuite nous pourrons utiliser notre nouvel utilitaire depuis une connexion telnet sur le routeur.

ATTENTION
Toute modification du firmware est potentiellement dangereuse.
Vous pouvez, en cas de mauvaise manipulation, perdre l'usage de votre routeur.
En cas d'utilisation d'un firmware custom, Netgear ne supportera plus votre matériel.
Vous êtes avertis.

Nous venons de voir le principe de modification du filesystem du DG834(G).
Avec cet exemple, certes simple, vous avez les bases pour l'ajout de fonctionnalité sur le routeur. Bien sur, le programme ether-wake ajouté seul, ne présente pas beaucoup d'intérrêt.
Pour finaliser, une modification des menus HTML du routeur, ainsi que le developpement d'un programme CGI sont nécéssaire pour une pleine intégration de la nouvelle fonctionnalité WoL dans l'interface d'administration du routeur.
A vous de jouer !

Un dernier conseil: si vous souhaitez vous lancer dans la création d'un firmware custom, la première chose à implémenter, c'est une busybox avec l'applet 'wget', ainsi vous pourrez tester vos nouveaux binaires en les téléchargent directement depuis le DG, dans son repertoire /tmp.
Vous évitant ainsi de reflasher à chaque essai ;).

Dans la 3eme partie de ce mini tuto, j'aborderai la recompilation du noyau linux.

@+',

Modification d"un firmware pour le DG834G --Premiere partie--

publié par malphx le dimanche, novembre 11 2007 - 22:52

Matériel requis

- Une machine Linux

Software requis

Une toolchain Linux MIPS Pour ma part, jchr(146)'utilise la toolchain fournie par Linksys pour son routeur WRT54g,que vous pourrez trouver sur le site de Linksys, partie GPL code.

Les sources GPL du firmwardisponible ici

Lchr(146)'utilitaire dgfirmware (créé par les membres du projet OpenDG834) Disponible ici en version source.

Installation de la toolchain

Pourquoi une toolchain MIPS ?

Cchr(146)'est très simple, le DG834G est architecturé autour dchr(146)'un System On Chip (SoC) de la famille AR7 de Texas Instrument.
Le processeur de ce SoC est un MIPS 4KEc cadencé à 150Mhz.Il nous faut donc une toolchain permettant de générer des binaires dans un format reconnu par ce processeur.
Nous allons grâce à cette toolchain, cross-compiler pour le DG834G. C'est à dire compiler des sources sur un type d'architecture machine (Intel x86 pour la plupart je pense), mais générer en sortie des executables pour architecture différente, en l'occurence Mips LE. (LE pour Little Endian).

Pour la toolchain, deux possibilités:

1. Utiliser une toolchain déjà construite :D
2. Eh non on va le faire nous même

Vous trouverez plus bas une description de lchr(146)'installation dchr(146)'une toolchain prête à l'emploi, puisque cchr(146)'est le choix que j'ai fais.

IMPORTANT: Cet article ne décrit pas la construction d'une toolchain MIPS.
Si vous cherchez des informations sur la construction => Google est votre ami !

Note: la description qui suit est faite à partir de lchr(146)'archive wrt54g.2.02.7.tgz provenant du site Linksys (lien plus haut) .

En avant pour lchr(146)'install !Une fois le tarball téléchargé:

DecompressionOn obtient le repertoire **WRT54G**

Copie de la toolchain sous **/opt**Elle se trouve dans le repertoire **WRT54G/tools/brcm**il faut copier **brcm** vers /opt

Modification de la variable $PATHIl faut ajouter les chemins: **/opt/brcm/hndtools-mipsel-linux/bin** et **/opt/brcm/hndtools-mipsel-uclibc/bin** à la variable **PATH**.

Création de liens symboliquesOn va créer des liens symboliques pour que les fichiers contenus sous /opt/brcm/hndtools-mipsel-uclibc/bin soit de la forme : **mipsel-uclinux-***
Pour cela, il faut depuis le repertoire /opt/brcm/hndtools-mipsel-uclibc/bin, lancer cette commande:

• `for i in ''ls''; do ln -s $i ${i/uclibc/uclinux}; done`Le résultat final pour ce repertoire:`[root@evectra bin]# lsmipsel-uclibc-addr2line mipsel-uclibc-g++ mipsel-uclibc-objcopy mipsel-uclinux-addr2line mipsel-uclinux-g++ mipsel-uclinux-objcopymipsel-uclibc-ar mipsel-uclibc-gasp mipsel-uclibc-objdump mipsel-uclinux-ar mipsel-uclinux-gasp mipsel-uclinux-objdumpmipsel-uclibc-as mipsel-uclibc-gcc mipsel-uclibc-ranlib mipsel-uclinux-as mipsel-uclinux-gcc mipsel-uclinux-ranlibmipsel-uclibc-c++ mipsel-uclibc-ld mipsel-uclibc-size mipsel-uclinux-c++ mipsel-uclinux-ld mipsel-uclinux-sizemipsel-uclibc-cc mipsel-uclibc-ldd mipsel-uclibc-strings mipsel-uclinux-cc mipsel-uclinux-ldd mipsel-uclinux-stringsmipsel-uclibc-cpp mipsel-uclibc-nm mipsel-uclibc-strip mipsel-uclinux-cpp mipsel-uclinux-nm`Ainsi la toolchain est prête à lchr(146)'emploi .

Les sources du firmware Une fois téléchargée et décompressée, lchr(146)'archive nous livre ses secrets:

```
[root@evectra V2.10.17_src]# lsapps build.sh
DG834B_V2.10.17.img linux-2.4.17_mvl21 patch-apps patch-knl
README target.tar.bz2 tools uClibc-0.9.19
```

Petite description de lchr(146)'ensemble:
apps: repertoire contenant les diverses applications opensource utilisée dans le firmware du DG.
build.sh: script permettant la reconstruction du firmware.
DG834B_V2.10.17.img: Image binaire du firmware (identique à celle que vous telecharger sur le routeur pour faire une mise à jour).
linux-2.4.17_mvl21: repertoire contenant les sources du noyau Linux.
patch-apps et **patch-knl**: respectivement, un patch à appliquer sur les sources des applications et un patch à appliquer sur les sources du noyau.
target.tar.bz2: Dans cette archive se trouve le **root fs**, une fois decompressé un repertoire **target**.
tools: repertoire contenant les outils permettant la création de l'image binaire:

- 7zip: Utilitaire de compression, utilisé au moment de la compilation du kernel (compression lzma).
- lzma: Utilitaire de compression, utilisé lors de la compression du filesystem SquashFS.
- mksqlfs: Utilitaire permettant la création de l'image binaire, remplace dans l'image du firmware la partie rootfs et y ajoutant le nouveau filesystem.
- sqlfs2tgz: Utilitaire permettant la transformation du filesystem compressé SquashFS en target en SquashFS 2.

uClibc-0.9.19: repertoire contenant la librairie C (version source).

Note importante
A ma connaissance, seule les sources du firmware 2.10.17 fournissent tous les utilitaires permettant de modifier nchr(146)'importe quelle partie du firmware (Kernel et rootfs).
Dans les versions précédentes, lchr(146)'utilitaire 7zip permettant la compression du kernel nchr(146)'était pas fournit, et il semble que ce soit une version exotique de ce dernier.
En effet, le 7zip officiel ne permet pas dchr(146)'obtenir un kernel valide.

Pour en terminer avec la préparation de lchr(146)'environnement de travail, il ne reste plus qu'à decompresser lchr(146)'archive **target.tar.bz2**

```
tar jxvf target.tar.bz2
```

On obtient ainsi le repertoire target, qui va nous permettre d'ajouter/retirer ce que nous voulons au rootfs.
Voila, cchr(146)'est la fin de ce premier article.
Notre machine est prête pour commencer le travail.
Dans la seconde partie, je traiterai un exemple de modification puis de reconstruction de l'image du firmware en vue de son téléchargement sur ce bon vieux routeur. ;)
@+