

Integrating DMA attacks in Metasploit

Rory Breuk rbreuk@os3.nl
Albert Spruyt aspruyt@os3.nl

University of Amsterdam

May 23, 2012

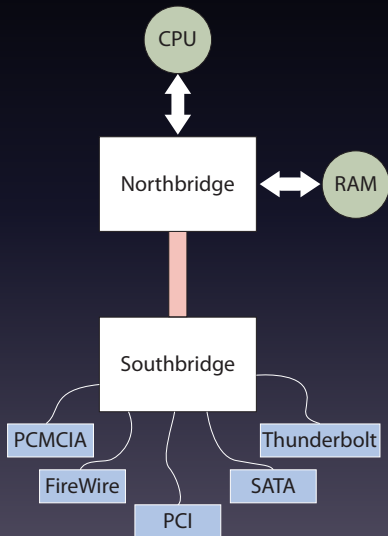


Introduction

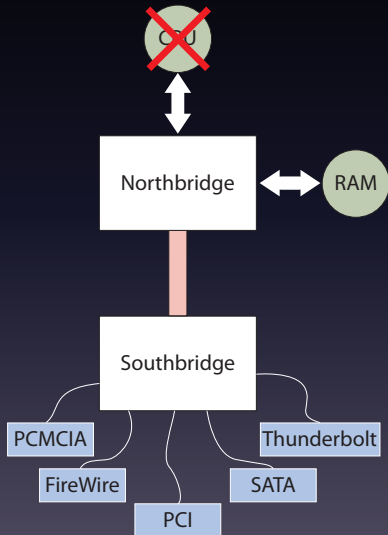
Goal:
Metasploit Over Firewire Ownage



Computer architecture



Computer architecture



Computer architecture cont.

Memory divided into 4KiB pages

Virtual / physical addresses

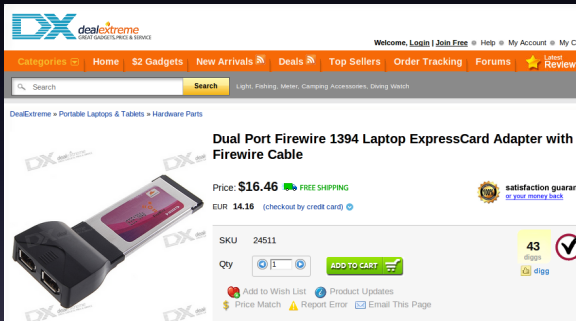


DMA attack vectors

FireWire

Thunderbolt

PCMCIA/CardBus/
ExpressCard



The screenshot shows a product listing on the DealExtreme website. The product is a "Dual Port Firewire 1394 Laptop ExpressCard Adapter with Firewire Cable". The price is listed as \$16.46 with free shipping, and a EUR 14.16 price is also shown. The product has a SKU of 24511 and a quantity of 1. There are buttons for "ADD TO CART", "Add to Wish List", "Product Updates", "Price Match", "Report Error", and "Email This Page". A "satisfaction guarantee" badge is also visible.

- Plug-and-Play and no driver required

Previous work

Encryption key/ password extraction
Winlockpwn/FTWAutopwn/Inception
libforensic1394



Goals

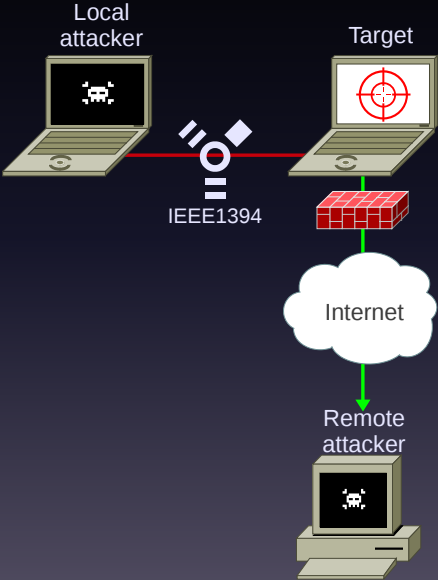
Use DMA attacks with Metasploit

Why?

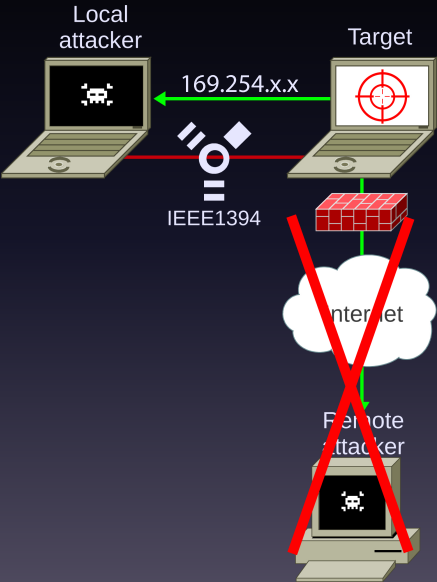
- Huge potential, but under utilized
- Widespread awareness is lacking
- Making it easy
- Lots of possibilities



Usecase

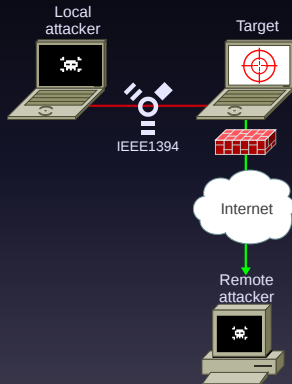


Usecase



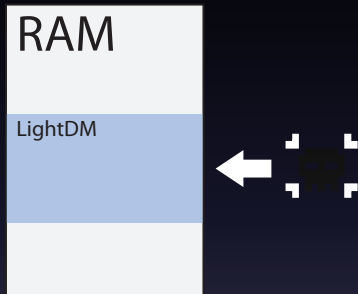
Metasploit concepts

Exploits
Payloads



Payloads

What to patch



Library call	<code>.text:0805C318</code>	<code>mov</code>	<code>eax, ds:dword_80705D8</code>
	<code>.text:0805C31D</code>	<code>mov</code>	<code>[esp+0FCh+endptr], 0</code>
Patch	<code>.text:0805C325</code>	<code>mov</code>	<code>[esp+0FCh+fd], eax</code>
	<code>.text:0805C328</code>	<code>call</code>	<code>_pam_authenticate</code>
	<code>.text:0805C32D</code>	<code>mov</code>	<code>[esp+0FCh+endptr], 2</code>
	<code>.text:0805C335</code>	<code>mov</code>	<code>[esp+0FCh+var_34], eax</code>
	<code>.text:0805C33C</code>	<code>lea</code>	<code>eax, [esp+0FCh+var_24]</code>
	<code>.text:0805C343</code>	<code>mov</code>	<code>[esp+0FCh+envp], eax</code>
	<code>.text:0805C347</code>	<code>mov</code>	<code>eax, ds:dword_80705D8</code>

Windows DEMO

Target: Windows 7 SP1 32bit

Find the signature

Inject payload



Problems



Need to interact with the system

Easily user detectable

Detectable by tripwire

Proposed solution

Stage 1:

- Inject stager
- Allocate new page

Stage 2:

- Restore originally patched code

Stage 3:

- Inject second stager
- Restore process
- Execute payload



Stage 1: Inject stager

Find signature

Save code

Inject special stager

	BITS 32
Save state	<code>call SAVEIP</code> <code>SAVEIP:</code> <code>pusha</code>
Allocate page	<code>xor eax,eax</code> <code>mov al,192</code> <code>xor ebx,ebx</code> <code>xor ecx,ecx</code> <code>mov ch,0x80</code> <code>cdq</code> <code>mov dl,0x7</code> <code>mov esi,0x22</code> <code>int 0x80</code>
	<code>shr ecx,1</code> <code>add eax,ecx</code>
Copy loop	<code>mov word [eax], 0xe0ff</code>
Jump to page	<code>jmp eax</code>

Stage 2: Restore code

Find the new page

Restore patched code



Stage 3: Finish

Upload second
stager + payload

*Directly overwrites
running code*

	BITS 32
Fork	<pre>mov ecx,eax xor eax,eax inc eax inc eax int 0x80 test eax,eax jz child</pre>
Restore process	<pre>parent: popa sub dword [esp],5 ret</pre>
Execute payload	<pre>child: mov esp,ecx sub esp,2048</pre>

Interactionless exploit

Xorg

- root permissions
- runs periodically



Linux DEMO

Target: Ubuntu 12.04

Look ma, no hands!

Stagers, IDS evasion

Target process is kept alive



Mitigation: theoretical

Theoretical:

- IOMMU

No practical implementations



Mitigation: practical

For the consultants:

- Don't buy them
- Destroy them / glue them
- Disable them
- Deny physical access

Does not guarantee safety



Achievements

Ported libforensic1394 bindings to Ruby
Integrate FireWire exploit into Metasploit
Reusable technique for DMA exploitation



Achievements

Enhanced attack:

- Smaller attack window
- Attack continued over TCP/IP
- Interactionless payload execution
- Use Metasploit functionality

<https://github.com/mrbreaker/mofo>



Metasploit Over Firewire Ownage

Questions?

