Project Maux Mk.II

“I Own the NIC, now I want a shell!”

Arrigo Triulzi
arrigo@sevenseas.org
“I have a Cunning Plan”

- Background concepts
- Last year's work (NIC takeover)
- Evolving Project Maux: Project Maux Mk.II
- Identification and defence
- Future work
How to read these slides

This is not a funded project but personal curiosity-driven what-if research,

Conceptually similar to the “nth Country Experiment” at LLNL in the 1960s: given open literature how quickly can two Physics PhD candidates develop a working nuke?

Given no prior knowledge, “the Internet”, a cheap 10-pack of NICs and a PC can we develop the ultimate rootkit?
Background concepts

NICs are becoming more intelligent:
- firmware is becoming more sophisticated,
- bugs creep into firmware leading to updates,
- updates on a deployed card are desirable (especially WiFi...).

So there must be a firmware loader...

© Arrigo Triulzi 2008
Background concepts

Video cards are becoming smarter:

- gamers want a “better experience” therefore accelerated graphics,
- the GPUs are becoming sophisticated,
- there is plenty of RAM to play with,
- a vendor ships a development toolkit...

A particular PCI NIC (Broadcom “Tigon”-based):

- MIPS CPU, very little on-board RAM,
- SDK available on web,
- Modified Linux drivers courtesy of CERN,
- Cheap 10-pack available.

© Arrigo Triulzi 2008

- SDK lead to development of “alternative” firmware (aka “Project Maux”):
  - hook to IP checksum routines,
  - add “sniffer” which copies packets into scratch RAM (circular buffer design).
- Only approximately 5s–30s sniffing on loaded NIC then circular buffer fills up.

Major issues with project Maux:

- lots of NICs blown in the process,
- the circular buffer is ridiculously small,
- loader requires Ring 0,
- sniffing alone is not very useful.
So what?

- A bit like a $^{235}$U device: easy to build but required lots of hardware,
- an excellent proof-of-concept strongly suggesting that the firmware avenue was worthwhile,
- it is an “obvious” entry point into the firmware hacking scenario,
- not that many players cover most hardware (Broadcom, RTL, Intel, Via, etc.).
Defences

Security through obscurity? Forget it, Google will (eventually) find it,

Assume your firmware SDK is in hostile hands,

Think about drivers sanity-checking hardware (need Secure Computing to really work)...

...but don’t assume PKI will save you...

© Arrigo Triulzi 2008
Evolving Project Maux

- We cannot fix anything at all on the NIC!
- Blowing NICs is an inevitable consequence of not having documentation,
- We can’t increase the RAM on the MIPS embedded controller,
- The loader only runs in Ring 0,
- No RAM means no extra functionality.
Evolving Project Maux

- Look elsewhere given the following constraints:
  - No OS support required,
  - Almost invisible to the CPU,
  - "Remote shell" capability,
  - Must be stealthy!

- This is where we go for the $^{239}$Pu design.

© Arrigo Triulzi 2008
Project Maux Mk.II

- The solution is to look at another PCI board!
- nVidia GPU development kit:
  - not the “OpenGL/Direct X” stuff...
  - but the “GPU computing” kit (aka “CUDA”)!
- This gives us access to a CPU with substantial amounts of RAM (128Mb at least)...

© Arrigo Triulzi 2008
Project Maux Mk.II

Overall design outline:

 quasi-SSH communications with quasi-SSH daemon running in GPU,

 NIC filters out “relevant” packets, forwards them to GPU via PCI-to-PCI transfer,

 quasi-SSH interprets them.
Project Maux Mk.II

- **Firmware**: Maux
- **NIC**: Hook IP checksum
- **PCI-to-PCI**: quasi-SSH in GPU
- **OS driver**: GPU code
- **GPU**: PCI bus
Project Maux Mk.II

NIC firmware modification:

- same technique as old “sniffer”: hook IP checksum (every IP packet triggers it),
- grab packet, check magic,
- pass “magic packet” to GPU via PCI-to-PCI transfer
check magic:

- if the IP ID is 0xbeef and
- the IP timestamp option has a flag value of 0x3, IP address of 0x50b1463d and a timestamp of 0x06026860.

this causes the firmware to forward the packet off to the GPU.
Project Maux Mk.II

GPU gets magic packet:

1st packet seen from this IP? Then send back a “suitable” response to say we are a “Mauxed” system with details of OS and capabilities,

otherwise interpret as part of a session.
Project Maux Mk.II

Introducing nicssh 1.0 (a quasi-SSH daemon):

- no DH key exchange, in fact no authentication!
- Blowfish with static 128-bit key (static as in “static in the GPU code being injected”),
- basic command shell with readline and limited number of commands.

© Arrigo Triulzi 2008
nicssh 1.0

- nicssh handshake:
  - ICMP Echo Request with “magic”,
  - respond with correct ICMP Echo Reply, but with “magic” in the header,
  - nicssh waits for 1st session packet.

- Note: the OS is totally oblivious to the above and never sees the ICMP packets.

© Arrigo Triulzi 2008
nicssh 1.0

nicssh capabilities:

- memory inspection (GPU RAM and system RAM),
- sniffer on NIC sending data to VRAM,
- sending of data via the network,
- cleanup *(extremely flaky)*,
- readline *(tab completion and history)*.

© Arrigo Triulzi 2008
nicssh 1.0

“Stealth” capabilities:

- if negotiated then use special port for traffic, otherwise default is 80,
- when using “web ports” (pre-defined to 80, 8080, 3128) then use rwwwshell GET method.

- More planned (Nushu, sniffing backdoor...).
nicssh 1.0

first “magic packet”

```
archimede:~/nicssh$ nicssh 10.4.4.233
Connecting to 10.4.4.233
ICMP Echo Reply from OS - no nicfw
archimede:~/nicssh$ nicssh 10.4.4.234
Connecting to 10.4.4.234
ICMP Echo Reply from nicfw (Windows system)
Requesting tcp/80 with cloaking
nicssh> ?
help memory* sniff* send* reboot cleanup quit
nicssh>
```

“magic packet” back

“stealth” mode

© Arrigo Triulzi 2008

“shell” with basic help
Installation...

Good question with no good answer at the time (March 2008). Some ideas:

- fake driver update with phishing website to entice downloaders (think “enhanced driver for gamers”),
- virus with injection payload,
- infected Linux distribution.
Installation...

Now for something more esoteric:

- Broadcom firmware has traces of “remote update” functionality...
- Drive-by injection via WiFi, WiFi driver exploit, PCI-to-PCI into the NIC?
- The second of the two would be lethal for laptops.
Uninstall?

What if you wish to remove all traces of modification?

Sorry, no (smart) answer at the moment (March 2008) for the NIC.

GPU? Just reboot (cold boot perhaps).

Should drivers always inject fresh firmware?

© Arrigo Triulzi 2008
Is it worth it?

Short answer: definitely **not** in 2008.

In the longer term this is an ideal A-V evasion technique for bots:


- bot functionality **in VRAM** and **on GPU**! OS is pristine and irrelevant.

© Arrigo Triulzi 2008
Is it worth it?

What about virtualisation?

VM escape: working on something named the “Jedi packet trick”...

Smaller number of server NICs make it an attractive “market” from an ROI perspective,

Hypervisor is “Just Another OS”,

“A Hypervisor” allegedly runs “Not Linux”...
Is it worth it?

What about firewalls?

90+% of world’s firewalls run on the PC architecture and therefore...

extension of “Jedi packet trick” to NIC-to-NIC transfers!

This could also extend to IDS/IPS systems... one parser vulnerability, NIC takeover, game over!

© Arrigo Triulzi 2008
Is it worth it?

Get nastier: what about the highly integrated support chips on the motherboard?

MITM of PCI-to-PCI transfers,
crypto accelerators need data and key to be sent to them to “accelerate”...
grab data being written to SATA...

Why? ROI for theft looks promising: keylogger++
Is it worth it?

Two scenarios:

Government: exercise for the reader.

Criminal organisations:

Targeted attacks “for rent”,

“Transparent” bots,

Pre-loaded bots: how difficult is it to attack Dell’s pre-loaded image loader?
But we use PKI!

PKI structure (assumed):

- signed firmware with a public key embedded in “secure” area of the chip,
- offer semi-custom parts to OEMs,
- OEMs roll-out their own modified firmware,
- so you have multiple public keys on chip.

One question: how do you push a CRL?
PKI, an interesting note

In the previous PKI scenario the problem is obviously pushing a CRL off to end-users.

The scenario was thought up independently and then was discovered to match the concerns of a manufacturer...

Think back to the Nth Country Experiment... I might not be the only one thinking about it.
Identification

On the network?

Currently as good as your detection of rwwwshell,

Can be improved due to “magic” needed for NIC firmware to redirect to GPU.

On the system?

Anyone counting PCI-to-PCI transfers?
Defence

The marketing defence: JIT manufacturing means different cards in different PCs and "same model different chipset" making targeting extremely difficult,

Firmware verification during update (c.f. Intel’s microcode update vs. AMD microcode update),

Trusted computing if extended to cover firmware verification (see John Heasman).
Defence

Learn from OS/VM (now zVM) providing fine virtualisation services since 1968:

- true hardware-aided virtualisation,
- protection rings at the hypervisor level,
- See IBM’s research (Karger paper on alpha PALcode and virtualisation).

“A virtualisation so good you can virtualise it”

© Arrigo Triulzi 2008
Defence

- The Secure Computing Initiative: currently does not appear to cover firmware! Obviously needs extending...
- the PCI bus could well become the “new Internet” for malicious communications,
- check all firmware, not just the obvious.
- Interesting question: how do you boot safely?
Project Maux Mk.II unsolved issues

- Elegant installation process:
  - Currently by hand (!!), no automation...

- NIC long-term stability testing:
  - Never ran for more than a week.

- GPU code persistence:
  - Reboot kills us at the moment.
Project cost

“Another year of Sundays”:
- approximately 100 man/hrs,
- Project Maux was approx. 150 man/hrs,
- $0 hardware costs (Project Maux: $100),
- MVT: Google.

© Arrigo Triulzi 2008
Future work

- GPU code persistence by using John Heasman's ACPI/BIOS work?
- GPU code persistence using hidden sectors on disk loaded at boot via NIC firmware BIOS initialisation routine?
- More sophisticated nicsssh functionality (with authentication?)
Drivers (sometimes) assume hardware is badly designed, perhaps badly behaved, but not malicious...

NIC takeover followed by driver takeover...

Driver takeover means Ring 0...

Ring 0 means kernel...

Who says that there has to be a single TCP/IP stack in the kernel? “There is no packet here, you will let it through” 😊

You heard it here first™

© Arrigo Triulzi 2008
Thanks

- My family ∀ their ∞ patience while I play with my toys,
- Toby for keeping the hard questions coming,
- Maya for project naming,
- $C_8H_{10}N_4O_2$

**Spoiler:** the Nth Country Experiment nuke design by the PhDs would have gone "b00m!" and was indeed initially meant to be tested "for real" at the NTS to validate this.

© Arrigo Triulzi 2008
References


Paul A. Karger, “Performance and Security Lessons Learned from Virtualising the Alpha Processor”, ISCA ’07

© Arrigo Triulzi 2008
References


Opteron Exposed: Reverse Engineering AMD K8 Microcode Updates, http://www.securiteam.com/securityreviews/5FP0M1P1FO.html

Papers by John Heasman (ACPI, BIOS and PCI rootkits):


http://www.nextgenss.com/research/papers/Implementing_And_Detecting_A_PCI_Rootkit.pdf


© Arrigo Triulzi 2008