
ARP Attacks

arp-sk in action

Frédéric Raynal <pappy@miscmag.com>

Cédric Blancher <blancher@cartel-securite.fr>

—

November 15th 2002

- Introduction

 - ARP basics and LAN attacks

- arp-sk

 - Tool presentation

- ARP cache poisoning

 - ARP cache poisoning applications using arp-sk

Introduction : ARP protocol (1/2)

0	7	15	23	31
ident. hwa addr.		ident. log. addr.		
lng hwa addr.	lng log. addr.	code		
destination ...				
... hwa address		destination ...		
... log. address		source ...		
... hwa address				
source log address				

Introduction : ARP protocol (2/2)

- ➔ ARP (Address Resolution Protocol - RFC 826)
 - binding layer 2 (Ethernet) et layer 3 (IP) addresses
 - client sends a layer 2 broadcast request (who-has)
 - destination answers a layer 2 unicast frame (reply)
- ➔ ARP cache usage
- ➔ no link between layer 2 and layer 3 information

Attacks : MAC Spoofing

- ethernet source address spoofing inside ethernet frame
- targets switches « CAM tables » (layer 2 addresses)

➔ Pros

- ▶ traffic redirection

➔ Cons

- ▶ target does not receive packets anymore, but continues to emit
- ▶ conflicts inside switch tables
- ▶ not very stealthy

Attacks : ARP Spoofing

- who-has sent using broadcast
- answering instead of legitimate host with spoofed datas

➔ Pros

- ▶ traffic redirection
- ▶ no need to bother with switches

➔ Cons

- ▶ Incertain result depending on who replies first

Attacks : ARP cache poisoning (1/2)

- create/modify victim ARP cache entries
 - victim's packets are directly sent to attacker
- ➔ Create an entry
- ▶ unicast who-has messages (Ok with RFC)
 - ▶ unicast who-has with spoofed datas
- ➔ Modifying an entry
- ▶ ARP spoofing : reply with spoofed datas

Attacks : ARP cache poisoning (2/2)

➔ Pros

- ▶ traffic redirection
- ▶ administrators rarely monitor ARP stuff
- ▶ difficult to actively prevent it

➔ Cons

- ▶ easy to detect

- Introduction

 - ARP basics and LAN attacks

 - arp-sk

 - Tool presentation

- ARP cache poisoning

 - ARP cache poisoning applications using arp-sk

arp-sk (1/2)

➔ Why a new tool ?

- ▶ gather functionalities from different existing tools (arpspoof, macof, arping ...)
- ▶ **complete** ARP headers manipulation (both Ethernet **and** ARP)
- ▶ add new functionalities (mapping, promiscuous mode detection, ...)

➔ arp-sk structure (since version 0.99.0)

- ▶ library providing basics functions
- ▶ arp-sk binary
- ▶ modules (dynamic libs) describing « scénarii » (one module == one functionality)

arp-sk (2/2)

➔ arp-sk options

- ▶ classical ones : emission frequency (in s, μ s or random), frame number, interface...
- ▶ basic module : packet type (-w, -r), Ethernet addresses (-s, -d), addresses within ARP message (-S, -D - [IP][:MAC]), randomly generated addresses (Eth and ARP, -rand*)

➔ TODO

- ▶ add new modules
- ▶ porting to other OS (especially OpenBSD and Solaris)
- ▶ create a library to build « network scénarii »

- Introduction

 - ARP basics and LAN attacks

- arp-sk

 - Tool presentation

 - ARP cache poisoning

 - ARP cache poisoning applications using arp-sk

ARP cache updates

- ▶ Opportunistic behaviour
- ▶ Entries creation
- ▶ Entries modification
- ▶ Entries deletion
- ➔ Let's attack...

Parameters we can play with:

- ▶ Ethernet: source MAC address
- ▶ Ethernet: destination MAC address
- ▶ ARP: layer 2 source
- ▶ ARP: layer 2 destination
- ▶ ARP: layer 3 source
- ▶ ARP: layer 3 destination

Entries creation

- ▶ ARP request
- ▶ ARP reply (depends on the OS and ARP cache state)
- ▶ Gratuitous ARP
- ➔ Not very useful

Entries update

- ▶ ARP request
- ▶ ARP reply
- ▶ Gratuitous ARP
- ➔ Interesting hosts are usually cached (DNS, GW, etc.)

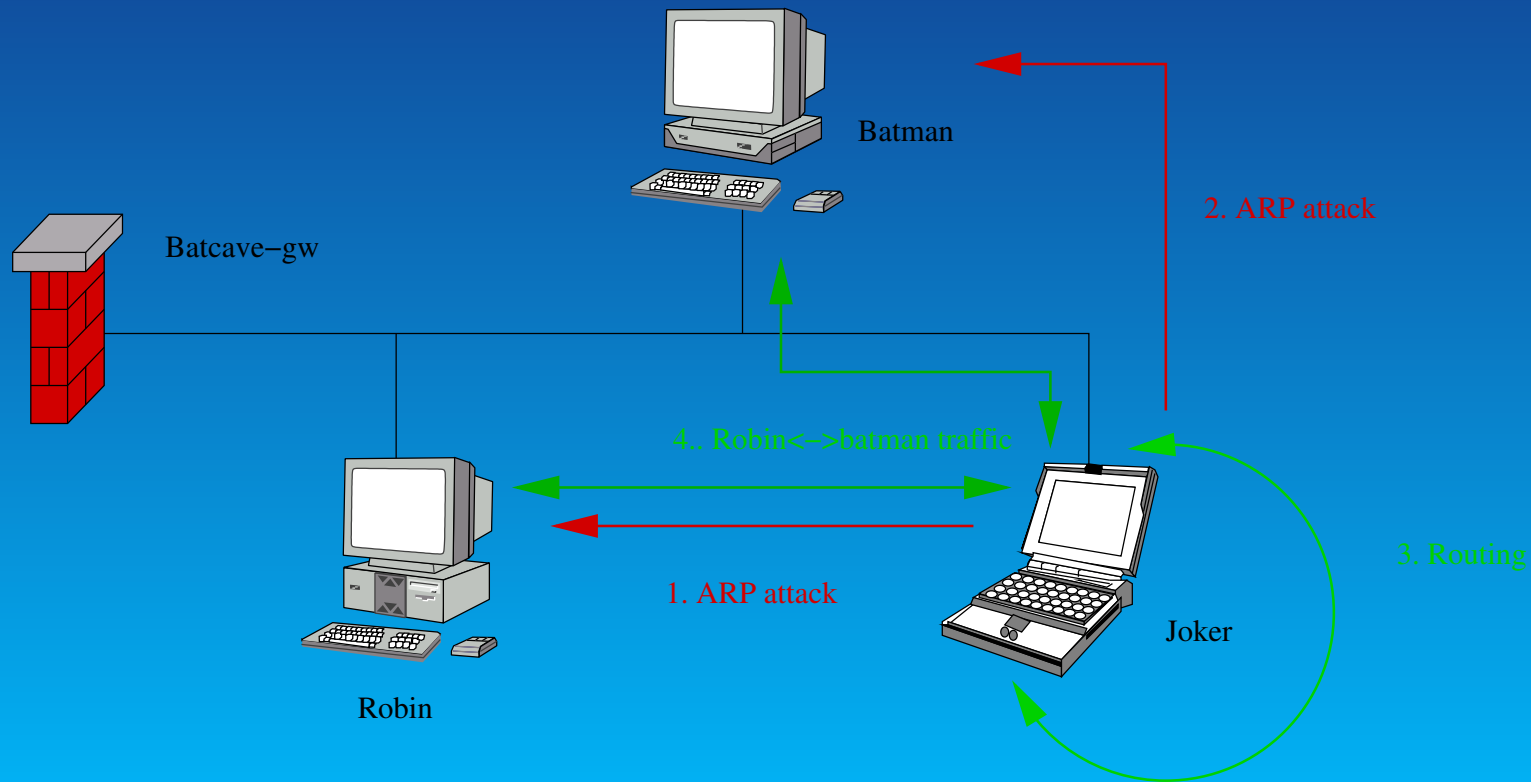
Entries deletion

- ▶ Entries expire
- ▶ ARP cache has a limited size (about 500 entries for Linux)
- ➔ We flood the cache, but we do not need to delete entries

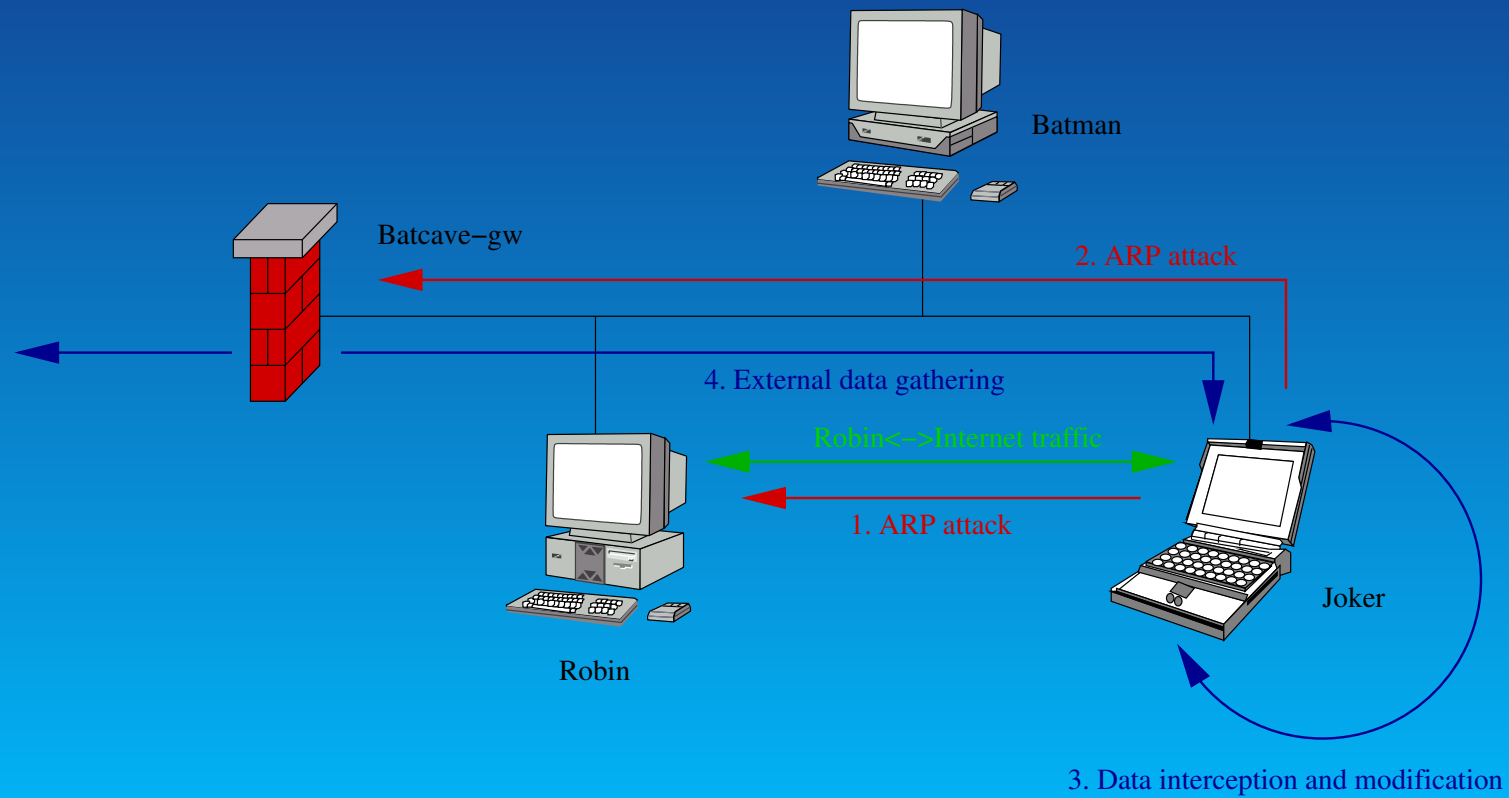
ARP cache poisoning applications

- ▶ Sniffing : we can listen to hosts traffic without using promiscuous mode
- ▶ Interception : we act as a transparent proxy for flows we intercept
- ▶ Modification : we can inject datas within proxied flows
- ▶ Hijacking : we can take one part's place within the flow
- ▶ Decrypting : classical MiM attack
- ▶ Spoofing : we can easily spoof another IP within the LAN
- ▶ DoS : flow destruction

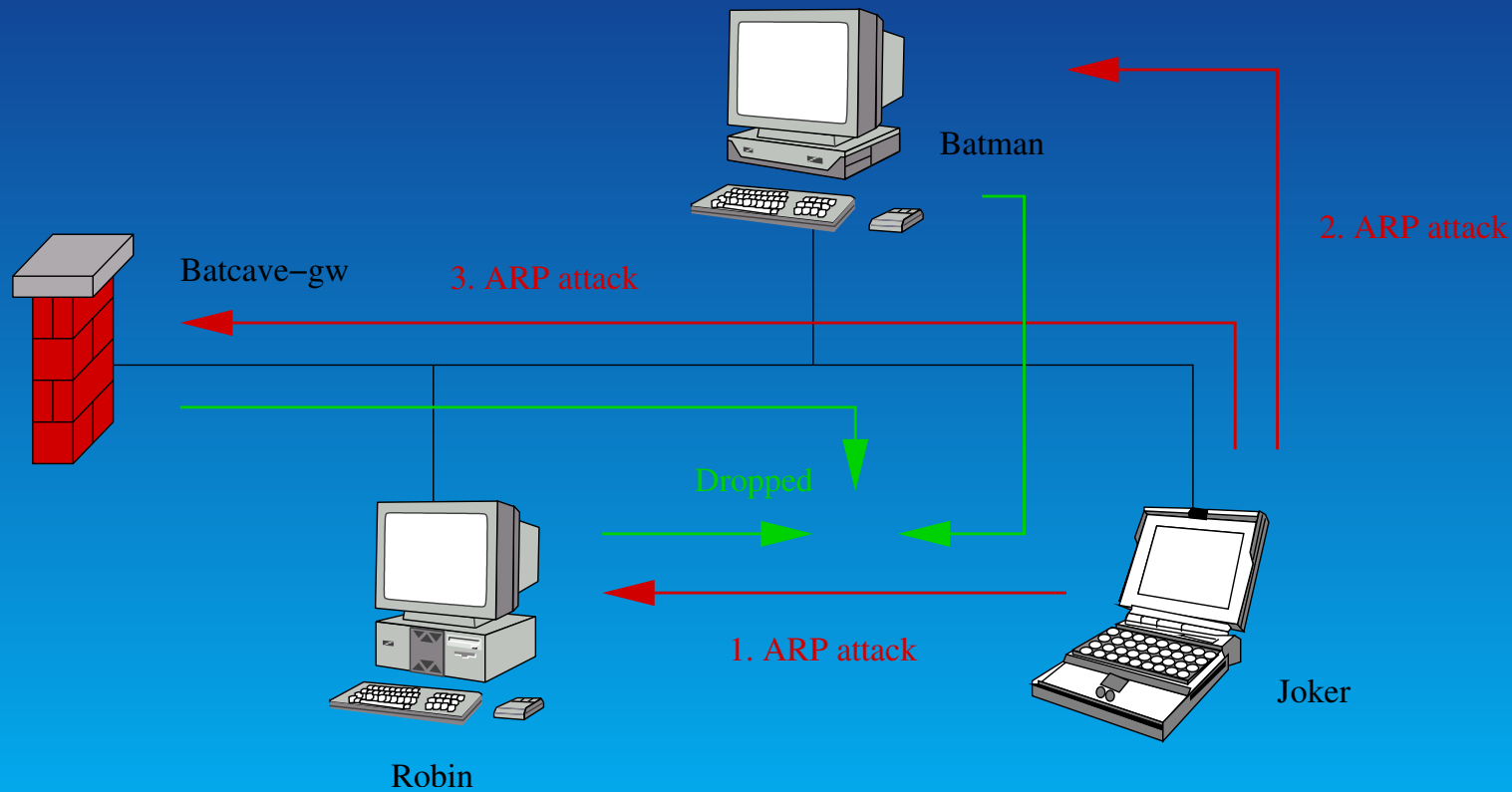
Sniffing with ARP MiM



Transparent proxying to intercept and tamper flows

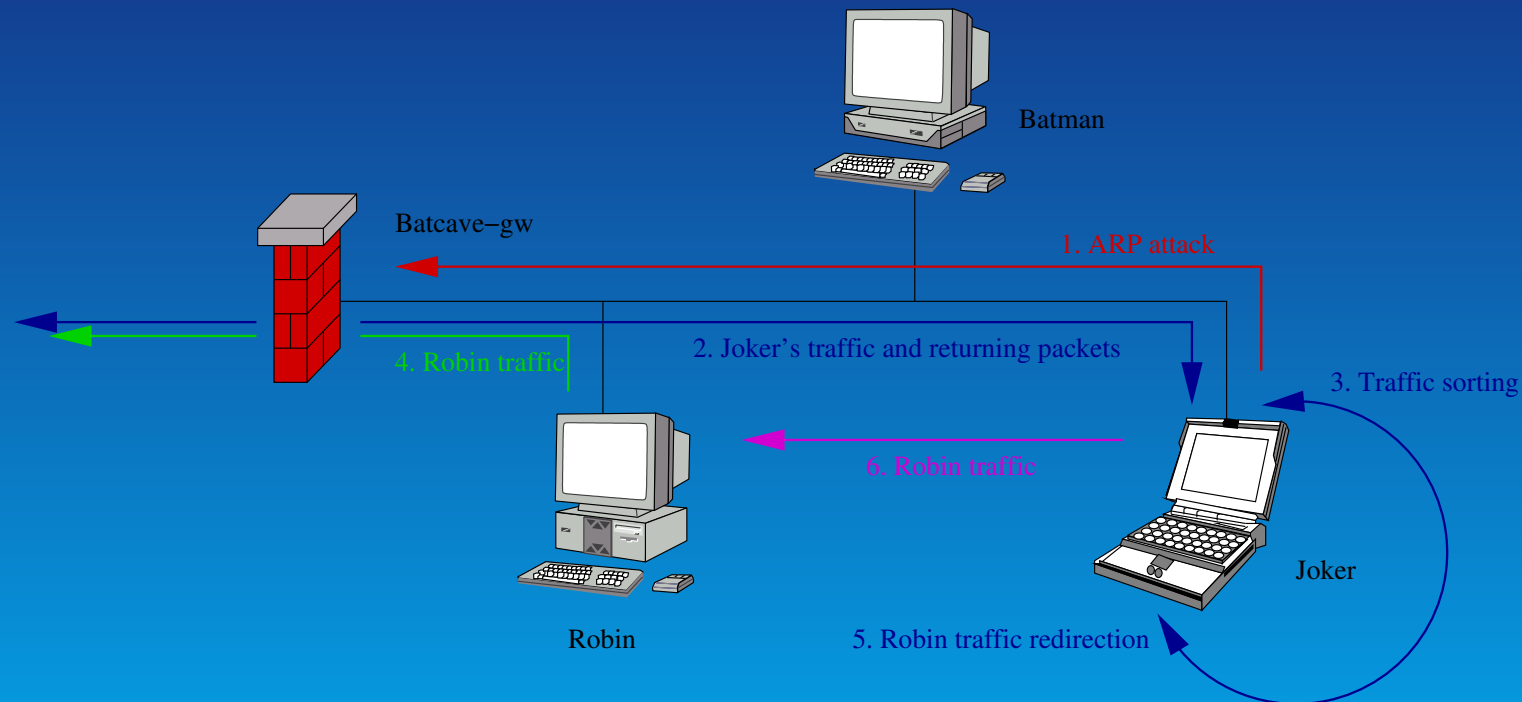


DoS



➡ Attacked hosts are likely to check their entries...

“Smart IP spoofing”



➔ We can also use ARP MiM ;)

➔ <http://www.althes.fr/ressources/avis/smartspoofing.htm>

Consequence

- ➔ Once an attacker gets root, he can attack the whole wire flows.

ARP is not secure and easy to fool : security was not in mind. We need stronger mechanism to enforce security :

- ▶ 802.1x

- ▶ Secure Link Layer

http://www.cs.wustl.edu/~fhunleth/projects/sll/sll_report.pdf

- ▶ Authentication within applications

Must be clear that switches are not security tools !

<PUB>



➔ MISC : french security magazine.

➔ <http://www.miscmag.com/>

</PUB>

- ▶ Éric Detoisien for writing winarp-sk and winarp-mim for Win32 platforms
- ▶ Laurent Licour and Vincent Royer for their experiments on OS behaviours