

Fighting Internet Worms With Honeypots

Laurent Oudot 2003-10-23

Summer 2003 will sadly remain famous for netsurfers because of the propagation of an Internet worm known as MSBlast, which infected millions of hosts running Microsoft Windows. This event is far from unique; other worms such as Slammer, Code Red, Nimda have similarly wreaked havoc in the past.

The goal of these roaming computer entities is to autonomously reproduce themselves on every reachable system on the Internet, resulting in ongoing problems with computer security.

The human tendency to use the same types of systems and applications, correlated to a kind of Darwinism theory for computer "monoculture", could have some security experts fear the widespread destruction of a given family of systems connected to Internet if an especially malicious new Internet worm were to appear. What would have happened if the recent worm MSBlast had formatted the hard drives of millions of infected Windows machines? It didn't, but it could have very easily.

As computer attacks evolve, new responses are essential.

This paper will evaluate the usefulness of using honeypots to fight Internet worms. The first part of the article will discuss some background information on worms and their ubiquity, then move on to discuss some of the interesting interactive functions of honeypots. Finally, we will study how a honeypot framework can be used to fight off Internet worms and even perform a counterattack, before we conclude with some future perspectives.

1.0 Introduction to Internet worms

Simply put, an Internet worm corresponds to mischievous code that spreads itself over networks [ref 1]. This dreaded entity usually attacks vulnerable hosts, infects them, and then uses them as means to bounce or propagate to other vulnerable targets.

Most of the time, worms are written and developed by computer hackers, researchers in the security field and virus authors. While viruses infections are essentially based on problems of abusing human vulnerabilities through social engineering, such as encouraging a user to click on an email attachment, Internet worms usually abuse technical vulnerabilities.

From the old functional standpoint from Edward Amoroso [ref 2], there are three primary actions of an Internet worm:

1. Infection : infect a target by exploiting a vulnerability.
2. Payload : launch malicious actions on the local infected target or toward others remote hosts.
3. Propagation : use the infected target as means for external propagation.

2.0 About honeypots

Honeypots [ref 3] are computer elements helping to delude aggressors. On a production network, evil hackers will attack some kind of fake system, losing time in doing so and giving information about themselves and their methods [ref 4].

When a honeypot is a dedicated host uniquely used to delude aggressors, it is supposed to play no role linked to systems in production. This implies that every request directed to the honeypot is suspect.

While honeypots are often thought to be used for passive analysis, they can also play an interactive role to deal with worms. Two kinds of honeypots are often used :

1. high interaction: a kind of real host is usually almost sacrificed (called a "sacrificial lamb") on a network while waiting for any aggressor.
2. low interaction: services and/or hosts are simulated (for example, Honeyd by Niels Provos).

3.0 Honeypots versus worms

The goal of this chapter will be to demonstrate the advantages of interactive honeypots to fight off Internet worms. While referring to the functional outline of the worms from the previous chapter, we will see how the technologies coming from the world of honeypots can be used during the three different phases of a worm: infection, payload, and propagation.

3.1 Honeypots and worm infections

The infection phase of a worm is when it abuses a vulnerability to copy itself onto a chosen target.

During this phase, the objective of a defender honeypot will be to detect abnormal behavior, such as aggressive flows of traffic, characterize it, then to send it to a dedicated network. This technique is usually called "bait and switch", and allows choosing what is supposed to reach the production network or the honeypots network.

The concept of a defender honeypot is based on a gateway, playing at once the role of a firewall and an intrusion detection system (IDS) or an intrusion prevention system (IPS). It filters the flow of traffic arriving to a network, analyses the content of the packets, and then evaluates if the source of a network session is malicious or not by comparing packets to signatures of known attacks. After this identification is carried out, the gateway will then be able to mark specific sources as dangerous for a chosen period of time. Thus, packets sent by marked sources will be redirected to the honeypot network rather than to the production network.

In the case of the recent worm MSBlast [ref 5], if the gateway matches incoming TCP packets with a

destination port 135 and has signs of the worm as determined by IDS signatures, it will be able to redirect them to a honeypot while marking the source as contaminated. The honeypot will then manage the future discussions coming from this new, freshly captured worm. We will see this interaction in section 3.3, the propagation chapter.

We can find some drawbacks with this technique :

1. Signatures are always a bit late for new, unknown aggressions.
2. Is this a reliable concept ? What could happen if the system makes errors and sends some legitimate traffic to the honeypot after a bad interpretation due to a false positive?
3. Network speed may slow down due to the huge work of packet analysis on the gateway.

3.2 Honeybots and the payloads of worms

The technologies around honeypots are very useful in dealing with payloads, especially to catch worms and analyze them. To catch a worm, you have to let it infect a host or at least let it think it is infecting a host. In the previous chapter we saw that we're able to divert such flows of aggression toward a honeypot.

This honeypot can either be a "sacrificial lamb" (a normal host without the very latest updates applied on, sacrificed in expectation of an attack), or just a simulation of services. Note that both kinds must not be able to bounce to other remote hosts which are not under your legal control. For more information, read the "data control" ideas proposed by the Honeybot Project.

Honeybots are valuable assets to help easily understand worms. For example, with MSBlast inside a honeypot we were able to simulate millions of hosts (potential targets) thanks to Honeyd with less than ten lines of configuration! Such a setup can be used to study the propagation speed using a large number of hosts without too much risk.

3.2.1 Sacrificial Lamb

When using a sacrificial lamb, a classical virgin computer can be installed with the required services on the appropriate operating system. This machine can also be part of a system accommodating several operating systems at once, through VMWare [[ref 6](#)].

To catch a worm, a defenseless machine is left to wait for an infection. Then it will be possible to try to look at the worm itself through the new binaries found on the disc and the captured network traffic.

When worms are made of only a small amount of code and are not very complex, it will be easy to analyze the network traffic and the binaries caught. But if the worm emits more complex packets at the network level (cipher), or if it modifies its behavior and eventually its signature (similar to viruses with polymorphism), the analysis becomes complicated. Moreover, if the worm quickly destroys infected hosts,

for example just after few bounces, the sacrificial lamb could be lost and all posterior analysis could be compromised.

3.2.2 Virtual hosts and services

By simulating hosts or services, a honeypot can try to dialog with remote incoming worms through the use of its fake services. The daemon called Honeyd [ref 7] is really sharp for such actions.

In order to illustrate this, here is a configuration allowing one to fool the recent MSBlast worm, making it think it had contaminated a true Windows host through the RPC DCOM service running over TCP port 135.

This simplified configuration was used with success to recover the MSBlast worm over the Internet:

```
create default set default personality "Windows XP Pro"
add default tcp port 135 open
add default tcp port 4444 "/bin/sh scripts/WormCatcher.sh $ipsrc $ipdst"
set default tcp action block
set default udp action block
```

And here is the content of our script "WormCatcher.sh" launched by Honeyd for every incoming request coming to TCP port 4444:

```
#!/bin/sh
# Creation of a directory for every contaminated host
# attacking the honeypot, in order to archive different binaries
mkdir /tmp/$1-$2
# Download of the worm through TFTP in this directory
# (specific behaviour for MSBlast)
cd /tmp/$1-$2/
tftp $1 <<EOF
get msblast.exe
quit
EOF
```

Thanks to this simple technique, we were able to recover binaries of the worm without even using a real Windows operating system! This technique is therefore rather practical, but it will be difficult to reproduce with worms that use more complex protocols and tools (such as a cipher with encoding keys for each network session). Until recently such types of evil worms have not been widely spread on the Internet.

3.3 Honeybots and the propagation of worms

3.3.1 Reply to the incoming requests of the worm

When a worm propagates itself, it often randomizes the IP of its future targets. In this list of targets, some IP addresses may be unused, others may not accommodate the remote services targeted by the worm, and some may not answer at all for other reasons. The worm will receive associated network errors and will then try to jump to other targets to spread itself as fast as possible.

While replying to a worm's incoming requests toward non-existing hosts or services, a honeybot will be able to steal time from the worm as it attacks its fake targets.

For example, MSBlast tries to attack the TCP port 135 of remote hosts that are randomly chosen. To reply to incoming requests on an unused IP address, turn on the demon called arpd (without specifying any argument) and configure Honeyd as follows:

```
create default
set default personality "Windows XP Pro"
add default tcp port 135 open
set default tcp action block
set default udp action block
```

Thanks to this configuration, if a MSBlast worm aims at an unused IP address from a network where Honeyd is running, it will see Honeyd's simulated hosts with a service opened on TCP port 135. The worm will believe it is a real and vulnerable host. With this technique, MSBlast is forced to attack the honeybot's fake hosts, losing time in the process and alarming administrators who will start to see many unusual packets. Administrators will also see where cleaning is needed through this activity, by identifying hosts that are under the worm's control.

It may not be easy to setup such a framework if a new worm uses network dialogs too complex to simulate, however this is not the case most of the time.

3.3.2 Reply slowly to the worm

Continuing the previous logic, a honeybot is able to see incoming requests toward nonexistent services or systems. Rather than just answering, it can also reply with specific, crafted packets and slow responses [ref 8] at the network layer [ref 9]. This technique has already been used with the tool called Labrea [ref 10]. This Unix daemon is able to reply with the unused IP addresses of a network, in order to simulate a TCP session with an aggressor. Then it slows down to maximize the duration of the TCP session, by using legal elements taken from the RFC. What actually happens is the TCP window size is permanently kept to 0,

preventing the worm attacking from sending data.

Usually a worm runs a process on the attacking host, using classic network calls of the kernel to spread itself elsewhere. By using the classic API in userland, such as sockets, a worm won't be able to understand why the network works slowly, and will be slowed down and blocked on many fake targets. If a worm designer wanted to handle that properly, the worm would probably operate slower (multiple checks would need to be added), become heavier (requiring more extensive code), and with stealth issues (low level access).

As an experiment, Labrea had been conceived to fight off the Code Red worm [ref 11] but it can sometimes deal with other worms as well. Note that Honeyd will support such features in future versions.

Nevertheless there are drawbacks to this defense system. For example, a multi-threaded worm, or a time aware worm, may simultaneously attack several targets, without remaining blocked on our fake targets.

3.3.3 Should we launch a countermeasure?

Beyond actions facilitating a slowdown, or indeed stopping a not-so-virulent worm, the possibility of blocking detected worm aggressions on the fly using a honeypot could also be envisioned. Thus, an action such as countermeasure, already known in the IDS world, will also be able to ease off or stop such aggressions.

Honeyd that are configured and specifically "warned" about a worm attack will be able to urgently ask for separation actions at the network layer. For example, on a LAN, by the bias of classical administration protocols (such as SNMP), a honeypot could itself ask a network switch to isolate a physical port of an infected attacking host while also creating an alarm on the security console.

Some other interesting countermeasure actions launched by honeypots could be:

- cutting off a network segment (a fast separation of flows to preserve the sensitiveness of important networks),
- isolating one or several hosts (by ports being shutdown on switches),
- closing remote, sensitive services used by the worm,
- inserting specific, crafted packets at the network layer to kill caught dialogs (though this is less useful using a honeypot than with an IDS),
- banning chosen network flows by inserting filtering rules on remote devices (routers, firewalls, etc),
- feeding a kind of active RBL (Realtime Blackhole List) dedicated to worms.

These techniques may suffer from certain problems, such as false positives that launch a huge shutdown of ports on internal switches. Moreover, honeypots are not used to deal with traffic that is not destined to them, whereas an IDS will handle every packet caught on the wire.

3.3.4 Launching a counter offensive

Countermeasures allow us to contain risks (host isolation), but this may not stop the problem of the worm itself. Honeypots can also be used to definitively regulate the very cause of the presence of worms on a network, as we will see hereafter, thanks to specific countermeasures considered "counter offensives" in this paper.

Here is the theory: we will suppose that there is a worm that has infected host A, and is trying to propagate itself on host B that is actually a honeypot. By looking at the attack coming from A, the honeypot can eventually assume that a worm succeeded in infecting host A, due to the technical vulnerability in that host. If this vulnerability was not removed by the worm's payload and host A is still accessible, the honeypot host B can launch a counter-attack on host A. In fact, by abusing the same vulnerability on host A that was used by the worm for its infection, host B can try to take the control of host A. If there is success, host B can try to kill the running process of the remote worm, clean host A, and then harden its security.

It is important to note that there are legal implications in doing this. The honeypot may not be allowed to fight the remote attacking hosts which are not under the legal control of an administrator. Despite the fact that a worm came from an external network, it remains illegal and is forbidden for one to fight back the attacking host accommodating the worm. Nevertheless, on a local network where an administrator has total legal control of all systems, such an architecture could allow blocking the propagation of a worm quickly.

In order to illustrate this concept, here is an example to limit the effectiveness of the worm MSBlast, thanks to Honeyd:

```
create default
set default personality "Windows XP Pro"
add default tcp port 135 open
add default tcp port 4444 "/bin/sh scripts/strikeback.sh $ipsrc"
set default tcp action block
set default udp action block
```

The TCP port 135 of the honeypot remains open and accepts remote RPC requests that abuse the DCOM vulnerability used by the MSBlast worm [ref 12]. Fortunately, this worm does not check the integrity of the RPC answers sent back to it, and thus having TCP port 135 open is sufficient because honeyd will answer the three way handshake and will acknowledge incoming evil packets.

Then in the configuration we add a service on TCP port 4444, which is the remote shell usually obtained and used by MSBlast to send its orders, including where to recover the full worm by TFTP and the execution of its downloaded code. This fake shell can be easily emulated by a script named "strikeback.sh" with the IP address of the incoming worm passed to it as an argument.

Here is the script called "strikeback.sh" used to launch the counter-attack against a MSBlast infected host. In this short example it is supposed to run under Windows XP Pro:

```
#!/bin/sh
# Launches a DCOM exploit toward the infected attacking host
# and then run cleaning commands in the remote DOS shell obtained
./dcom_exploit -d $1 << EOF
REM Executes the following orders on the host :
REM 1) Kill the running process MSBlast.exe
taskkill /f /im msblast.exe /t
REM 2) Eliminate the binary of the worm
del /f %SystemRoot%\system32\msblast.exe
REM 3) Clean the registry
echo Regedit4 > c: \cleanerMSB.reg
echo [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run] >> c:
\cleanerMSB.reg
echo "auto windows update" = "REM msblast.exe" >> c: \cleanerMSB.reg
regedit /s c: \cleanerMSB.reg
del /f c:\cleanerMSB.reg
REM N) Specific actions to update the Windows host could be added here
REM N+1) Reboot the host
shutdown -r -f -t 0 exit
EOF
```

This script, given strictly as an example, can be improved upon by using evolved programming languages such as VBS. A longer example [ref 13] has been tested on a research network, cleaning our infected hosts in a few minutes.

Some SysAdmins were recently polled to determine if it is ethical to take active defense measures in such a targeted, counter offensive way, within a network their organizations owns. The results can be seen here [ref 14, page 29 & 32] (76 respondents).

4.0 Conclusion

Technologies from the honeypot domain become an interesting card to play in the fight against Internet worms. They can be used to redirect evil worm traffic to dedicated fake services, safely catch the worms and analyze their behavior, and finally limit their propagation through networks.

These young technologies are therefore very promising, though they probably still suffer of a lack of testing

experience when used over wide networks.

In case of attacks coming from a very hostile worm (black worms) that can kill or pester the targets, and can protect themselves or understand if they are deluded, the use of honeypots may be rather limited without a strong technical analysis to understand what is done by the worm and how to play with it. By our luck, so far none of the known Internet worms have been so violent.

Without becoming the principal key allowing total lockdown of computer architectures, honeypots are a valuable additional means in the fight against the Internet worms. More technical details will be explained at Black Hat Asia 2003 [ref 15], showing how Honeypots may stop Internet Worms from wreaking havoc.

Credits

Thanks to Lance Spitzner for his ideas and all I have learned from him, and to Kelly Martin for his careful reviewing and help.

About the Author

Laurent OUDOT is a computer security engineer employed by the *Commissariat a l'Energie Atomique* in France. On his spare time, he is a member of the team *Rstack* with other security addicts. Concerning honeypots, Laurent is an active member of the *French HoneyNet Project* which is part of the *HoneyNet Alliance*.

References

- [ref 1] Ryan Permeh et Dale Coddington (Eeye), " Decoding and understanding Internet Worms ", 21st November 2001, <http://www.blackhat.com/presentations/bh-europe-01/dale-coddington/bh-europe-01-coddington.ppt>
- [ref 2] Edward Amoroso, chapter 4.5 from " Fundamentals of computer security technology ", explaining replication of the worms called "Internet viruses" ; see " Typical virus operation "
- [ref 3] Lance Spitzner, " Honeypots, tracking the hackers ", 2002, <http://www.tracking-hackers.com>
- [ref 4] " Cuckoo's egg ", the cult book (based on a real history about evil hacking and spy) that probably first talked about the concept of honeypots.
- [ref 5] Security advisory for the MSBlast worm that appeared on the 11th august 2003 and that abused a vulnerability announced in July 2003. <http://www.microsoft.com/security/incident/blast.asp>
- [ref 6] Very useful tool for the Honeypot community that can launch multiple instances of different operating systems at the same time on a same host. <http://www.vmware.com>

[ref 7] Niels Provos, " Honeyd, a virtual honeypot daemon ", 10th DFN-CERT Workshop, Hamburg, Germany, february 2003 <http://www.citi.umich.edu/u/provos/honeyd/>

[ref 8] Tony Bautts, " Slowing down Internet worms with tarpits ", 21th august 2003, <http://www.securityfocus.com/infocus/1723>

[ref 9] Zesheng Chen, Lixin Gao, Kevin Kwiat, " Modeling the spread of active worms "

[ref 10] Tom Liston, " Welcome to my tarpit, the tactical and strategic use of Labrea " <http://labrea.sourceforge.net/labrea-info.html>

[ref 11] CAIDA, " Caida Analysis of Code-Red " <http://www.caida.org/analysis/security/code-red/>

[ref 12] Security bulletin MS03-026 about the vulnerability of the RPC DCOM services, used by the MSBlast worm to infect remote hosts, http://www.microsoft.com/security/security_bulletins/ms03-026.asp

[ref 13] A VBS script given as an example to clean variants of the MSBlast worm under Windows NT, 2000 and XP <http://www.rstack.org/oudot/cleaner.vbs>

[ref 14] Active Defenses to Cyber Attacks, workshop 9/03, supported by a research grant from Cisco <http://staff.washington.edu/dittrich/ad/AD-workshop-091203.ppt>

[ref 15] "Honeybots Against Worms 101", Laurent Oudot, Black Hat Asia, December 2003, Singapore <http://www.blackhat.com/html/bh-asia-03/bh-asia-03-speakers.html#Laurent>

[Privacy Statement](#)

Copyright 2006, SecurityFocus