

HoneyPot Forensics Part I: Analyzing the Network

A major goal of honeypot research is to improve our knowledge of blackhats from two perspectives: *technical* and *ethnological*. For the former, we want new ways to discover rootkits, Trojans, and potential zero-day exploits (although capturing zero-day

building some aggregates to identify blackhat-generated backdoors and network traffic.

- *Know your system.* As soon as a blackhat compromises the honey-pot, he can explore the honey-pot's neighborhood. We can also bet he'll download a rootkit—if he didn't already notice he's in the trap and under surveillance.
- *Know your enemy.* Now that we've gathered network and system evidence, we can build what we call "the inner circle," which describes the links between the blackhat and the honey-pot. We can do this by correlating the events noticed in previous analysis.

Of course, some events will appear in both network and system analysis, but some will only be viewable on the network or the system. Thus, finding the intersection $N \cap S$ of two sets of events is important, with N being the set of network events and S the system events. All events might not appear in this intersection, though, so you will also need to look at the symmetric difference of those sets, $N \Delta S$, which shows the events detected solely through network or system analysis.

Even if we could consider the analysis done at this point (since the intrusion's complete scenario is revealed), it's worth taking a deeper look at the network flows and tools the intruder used: they can lead to other resources such as FTP or Web servers. Such an analysis, while not directly connected to the honey-pot, is very useful because it helps the analyst with the never-ending tasks of updating the intruder's profile and gathering more information.

exploits in a honey-pot is an unusual event). For the latter, we want a better understanding of the areas of interest and hidden links between blackhat teams.

One way to achieve these goals is to increase the verbosity of our honey-pot logs and traces so that we learn every single action the intruder made. The most common tools for doing this are Sebek (<http://project.honeynet.org/tools/sebek/>) for system events and Snort (www.snort.org) for network activity. Unfortunately, there is no easy way to correlate information from these sources, which complicates honey-pot forensics. Although computer forensics focuses on analyzing a system once we suspect it has been compromised, we expect honey-pots to be compromised. Thus, honey-pot forensics focuses on understanding the blackhat's techniques and tools, before and after its intrusion on the honey-pot.

Setup and context

The type and setup of a honey-pot and its environment are very important because they affect analysis. What do you expect to catch with the honey-pot you just deployed—nonpublic exploits and next-generation rootkits? You might succeed, but to catch a big fish, you need special

bait. Keep in mind that honey-pot technologies are still quite young, and that honey-pots themselves are easy to fingerprint. Even if a skilled blackhat falls into the trap you've prepared, he might realize where he is very quickly and leave immediately.

Knowing the honey-pot's context is the first step in forensics because it helps you understand what happened during the compromise. Forensics is not just a technical problem, it's a human challenge: the analyst must go beyond what is typically expected from a coroner. Although the coroner just deals with technical questions (where did the intruder come from, what rootkit did he or she use, and so on), the analyst's job requires the skills of a coroner as well as those of a detective. Here are some guiding principles:

- *Know your honey-pot.* Hosts connected directly to the Internet, located on a company's LAN, or made available over a wireless network won't catch identical bees. The level of defense built into the honey-pot to avoid abuse is also an important parameter to take into account.
- *Know your network.* A honey-pot is placed on a network and thus receives traffic before, during, and after a compromise. We must focus on network activity at a rather high level by looking for scans and

FREDERIC RAYNAL
MISC
Magazine

YANN BERTHIER
Hervé
Schauer
Consultant

PHILIPPE BIONDI
Arche/
Omnetica
Group

DANIELLE KAMINSKY
TEGAM
International

Network activity analysis

When dealing with a compromised host, the first sources of potentially available data are

- the network audit trail in the firewall logs,
- network flows as exported by the network gear, and
- full packet captures with inline or passive network sensors.

Forensic analysis of a honeypot generally doesn't differ from a similar analysis of compromised hosts, except for two crucial points: all traffic from the honeypot itself is interesting, and a full capture of traffic to and from the honeypot is usually available. This data availability is a consequence of the fact that on a honeypot, an intruder's arrival is very much anticipated. The same cannot be said for a compromised server, where isolating legitimate traffic from questionable traffic can be quite tricky: relatively few sites routinely record network transactions.

The big picture

As part of our honeypot's deployment, traffic going to and from our honeypot was recorded, which provided us with a roughly 19-Mbyte PCap file (www.tcpdump.org) containing 192,700 Ethernet frames recorded over 24 hours. This record contains all our intruder's moves, from the time of the intrusion to the use of our honeypot as an IRC bouncer. The first challenge facing a security analyst studying a network trace like this is to build a timeline of events based on packet traces. This timeline must then be correlated with the one built from system forensics.

Even if our 19-Mbyte data set seems small to most network auditing teams, honeypot-related network captures generally contain large numbers of packets. As a first step toward seeing the bigger picture, we begin with some basic rough statistics about the 192,700

frames we captured. First, we enumerate the layer-3 and 4 protocols, which gives us some Internet Message Control Protocol (ICMP), Transmission Control Protocol (TCP), and User Datagram Protocol (UDP) connections, but nothing else. This limits the probability of surprises: no Generic Routing Encapsulation (GRE) tunnels over IPv6, and no raw IP attacks. Ultimately, we end up with 143 ICMP packets and 41 UDP packets, but they don't reveal much: everything seems to be in the TCP connections.

The problem is that tracking these connections is painful with the usual tools, mainly because they're full of blackhat-initiated SYN scans (which stands for SYNchronize). One trick is to look solely at SYN-ACK (SYNchronize/ACKnowledge) packets, so that we only catch connections to open ports and avoid the large number of ignored (no response) or RST (which stands for ReSeT) packets. If we do this, we reveal 148 such packets, which are open connections and successfully scanned ports.

These open connections can be displayed as an oriented graph (see Figure 1), where each node is an IP attached to one or more ports to which a successful connection has been made. The direction of the edge is from the system that opens the connections (the one sending the SYN packet) to the system that acknowledges this packet (the one sending the SYN-ACK packet). We

graph for incoming connections and one for outgoing (relative to the honeypot), we get Figure 2. As the incoming connections graph shows, some connections to ports should be closed (ports 50, 31337, and 45295). This is clear proof of a compromise: only three hosts are connected to more than one port, with two of them connected to port 139 and port 45295. Moreover, some machines seem to connect directly to the Eleet port (31337) without any other connection.

The outgoing connections graph shows the blackhat connected to two mail servers, four IRC servers (one of which is also accessed via port 6660), one HTTP server, and three machines on the NetBIOS port. We can also see two interesting IPs accessed on the FTP port and a lot of high ports (those above 1024). We can deduce that four passive FTP data transfers have occurred on one server and eight on the other.

Ordering the chaos: How tools and techniques help

Network flows, as exported by network devices, were designed with billing and accounting in mind. However, because they're a cheap and efficient way to record network transactions, they're also valuable to the security practitioner for denial-of-service detection, tunnel detection, and other uses.

Although several network flow implementations exist, each with its

Once we isolate relevant flows in the context of our analysis, the next step is to pinpoint meaningful events among the records, which we can use to build a timeline.

quickly see that none of the boxes except the honeypot has both incoming and outgoing connections.

If we split the graph in two, one

own strengths and weaknesses, Cisco's NetFlow is the most widely used. Cisco devices can be configured to export NetFlow records to a collector,

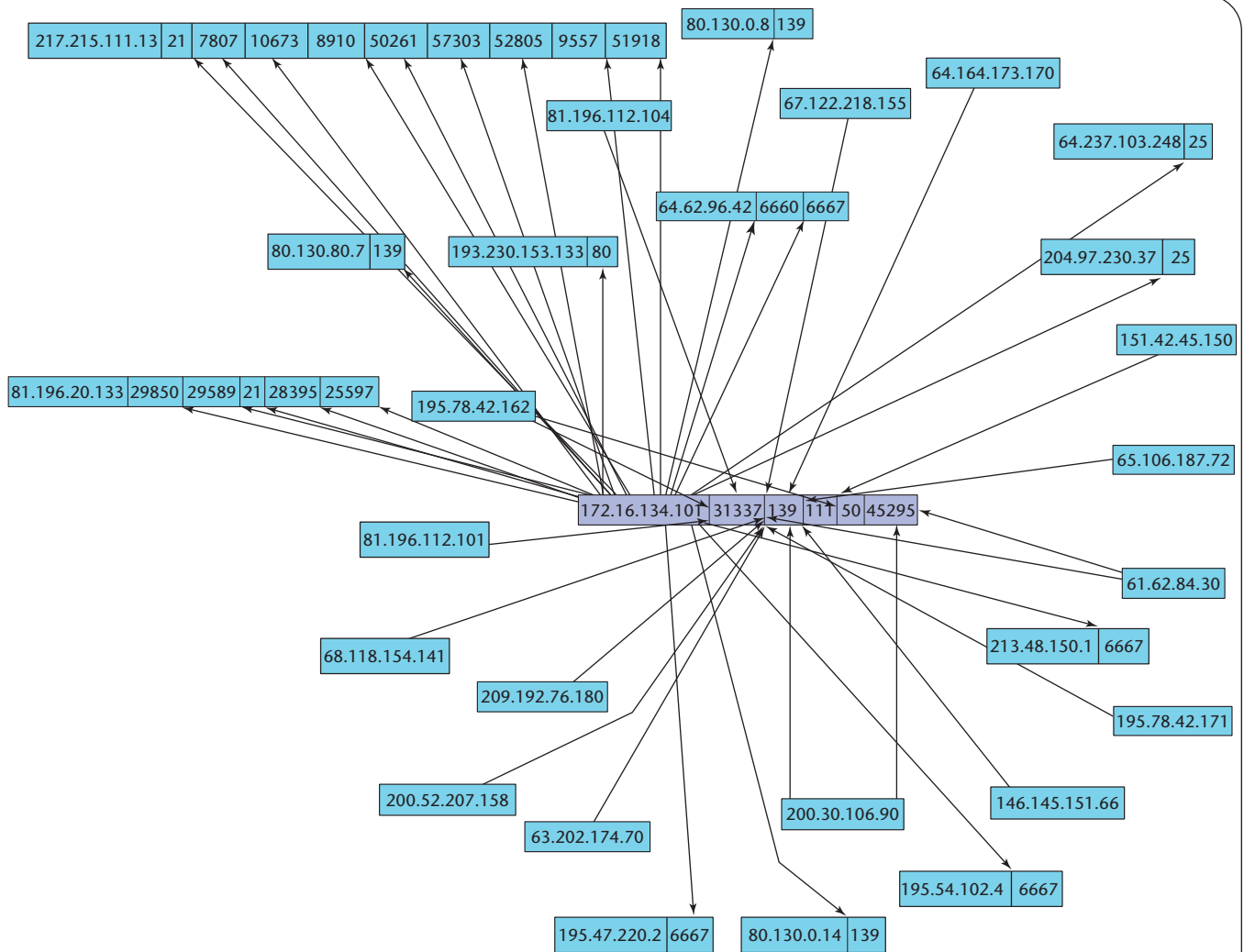


Figure 1. Graph of incoming and outgoing TCP connections. These connections are marked from the system opening the connection to the remote open port.

where several tools are available to store those records to disk and work on them later.

In our case, because we have an existing packet capture (and no exported NetFlow accounting data) and don't want to replay the traffic, we chose a tool that can work on an existing PCap capture. The open-source software program Argus (<http://qosient.com/argus>) can generate network flows from the PCap trace. Argus has two main parts:

- a server, which reads PCap data (either a live capture or an offline file) and exports Argus flow

records, and

- several specialized clients, which take Argus flow records or NetFlow data as input.

Argus implements its own flow record format, which differs from other implementations in two ways: Argus flows are bidirectional, and Argus maintains state information for network transactions.

After running Argus on our PCap trace, we had a complete dump of all network flows to and from our honeypot, with each record containing the following information:

- flow start and duration,
- protocol,
- source and destination IP,
- ports (when applicable),
- number of packets and bytes (sent and received), and
- transaction status.

Not all flow records are relevant for analysis: the Internet is a noisy place, where scans are common and usually happen as soon as a host plugs in. That's what happened to our honeypot, which was scanned continuously by many sources from the very second it went online. These noisy scans, which include flows from

sources not playing a role in the honeypot's compromise, can mark the blackhat's actions as irrelevant and dismiss them. Naturally, there's a possibility that, hidden in the dismissed noise, one of the scans originated from a machine controlled by an intruder during a preliminary discovery phase. This is pure speculation, though, because there's no way to confirm or deny it.

Once we isolate relevant flows in the context of our analysis, the next step is to pinpoint meaningful events among the records, which we can use to build a timeline. Using several Argus clients, we can aggregate and sort flows under different schemes:

- relatively long flows,
- flows in which many bytes have been exchanged,
- honeypot-originated flows, and
- subnet aggregation.

Whereas Argus will not give us any information about what was contained in all these network flows, it quickly provides a good estimate of network activity on the honeypot. For instance, massive scans on complete network classes appear immediately.

Building the network timeline

Our network capture started at 01:02:54 a.m. on 12 September and ended at 01:01:03 a.m. the next day. The two longest flows are as long as the capture itself (see Figure 3): the first one is a NetBIOS broadcast, which is due to the presence of a Samba daemon on the honeypot, and the second one, while not directly relevant, is the Sebek data sent to the Sebek server.

The first interesting flows appear in the capture a little before 6:00 a.m.: a series of 23 flows begin at 05:50:33 a.m. from an address located in the Asia Pacific Network Information Center (APNIC, www.apnic.net) zone to a not-well-known port of the honeypot, 45295/TCP

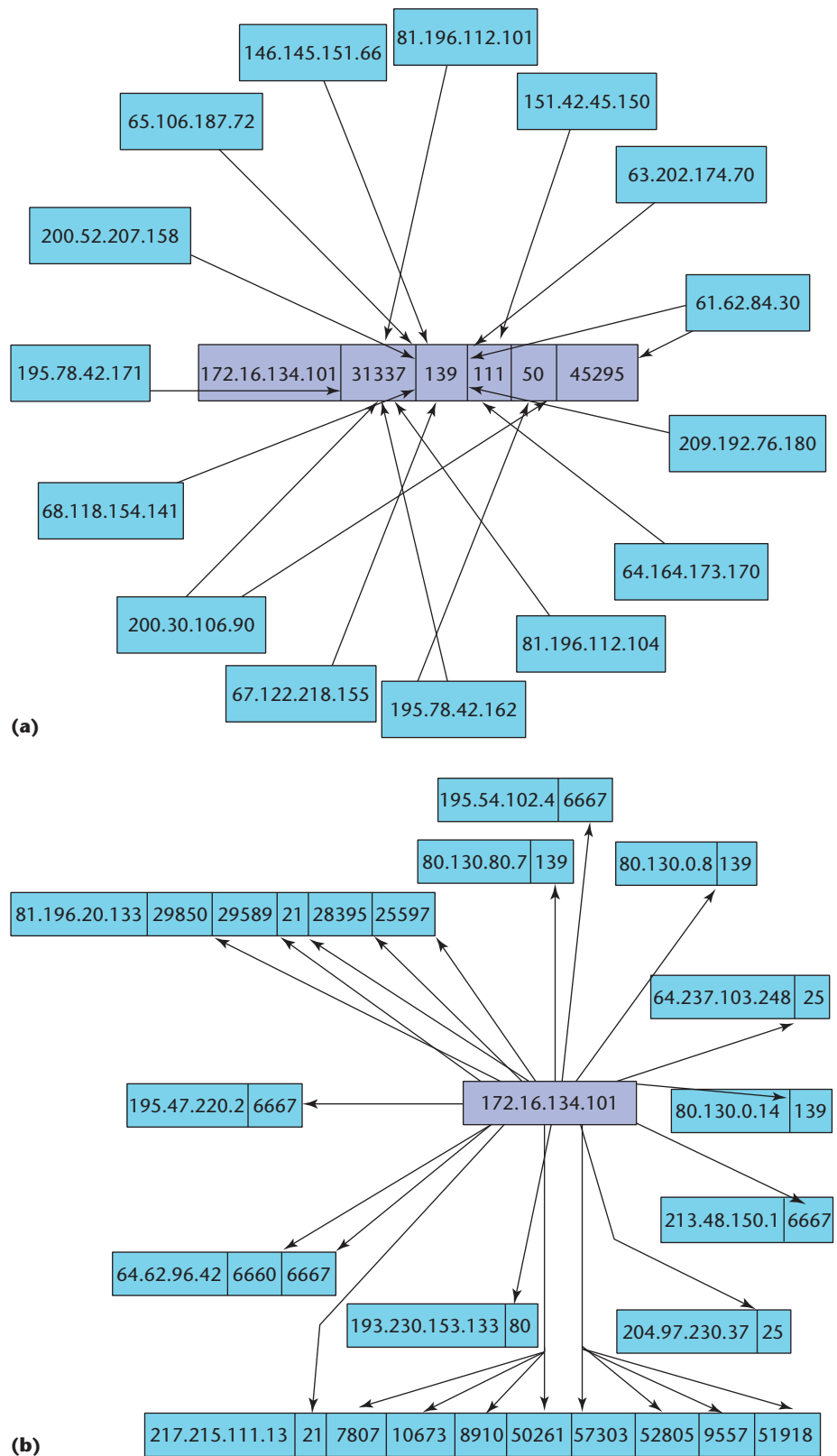


Figure 2. Graph of TCP connections. (a) Incoming TCP connections quickly reveal what reaches the honeypot, even to those ports that weren't supposed to be open. (b) Outgoing TCP connections show actions performed by the blackhat on the honeypot.

```
01:02:54 85685 17 172.16.134.101.138 → 172.16.134.127.138 238 0 60095 0 INT
01:32:48 79787 17 172.16.134.101.1101 → 10.0.0.1.1101 1825 0 172301 0 INT
```

Figure 3. Longest flows. Flow 1 is due to a NetBIOS server on the honeypot, whereas flow 2 corresponds to the logs sent to the Sebek server to supervise the blackhat.

```
05:50:33 0 6 xx.xx.84.30.42893 → 172.16.134.101.45295 1 1 74 54 RST
05:50:33 0 6 xx.xx.84.30.42895 → 172.16.134.101.45295 1 1 74 54 RST
05:50:33 0 6 xx.xx.84.30.42897 → 172.16.134.101.45295 1 1 74 54 RST
[ . . . ]
05:50:37 0 6 xx.xx.84.30.42938 → 172.16.134.101.45295 3 2 198 132 FIN
05:50:37 0 6 xx.xx.84.30.42939 → 172.16.134.101.45295 2 2 132 132 FIN
05:50:37 14479 6 xx.xx.84.30.42940 → 172.16.134.101.45295 4 2 272 140 FIN
05:50:39 0 6 xx.xx.84.30.42931 → 172.16.134.101.45295 2 2 132 132 FIN
05:50:39 0 6 xx.xx.84.30.42932 → 172.16.134.101.45295 2 2 132 132 FIN
05:50:40 0 6 xx.xx.84.30.42934 → 172.16.134.101.45295 3 3 210 206 FIN
```

Figure 4. First steps to the honeypot. The change from RST to FIN shows when someone starts to listen on port 45295 at 05:50:37.

```
05:44:21 1 6 xx.xx.84.30.42222 → 172.16.134.101.139 4 3 272 206 FIN
05:50:31 1 6 xx.xx.84.30.42254 → 172.16.134.101.139 2 2 4649 542 RST
05:50:32 1 6 xx.xx.84.30.42882 → 172.16.134.101.139 8 8 4649 612 RST
05:50:32 1 6 xx.xx.84.30.42883 → 172.16.134.101.139 8 7 4649 546 RST
```

Figure 5. Attacks on port 139. The person connected to port 45295 made previous attempts to connect to port 139.

```
15:29:54 0 6 xxx.xx.106.90.1902 → 172.16.134.101.45295 3 1 206 74 FIN
15:29:55 0 6 xxx.xx.106.90.1900 → 172.16.134.101.45295 1 1 74 74 TIM
15:29:56 96 6 xxx.xx.106.90.1901 → 172.16.134.101.45295 6 6 344 444 RST
15:29:57 219 6 xxx.xx.106.90.1905 →? 172.16.134.101.45295 32 33 2201 9996 FIN
```

Figure 6. Direct connection to port 45295. This is where the intruder enters the honeypot.

(see Figure 4). The first flows end with an RST after an exchange of two packets: this is because the attacker sends SYN's to a closed port. Suddenly, at 05:50:35 a.m., the flow ends with a FIN (FINish) packet, which suggests that a socket is bound on the port 45295/TCP, where no

service was listening before.

At this point, looking at flows previously considered irrelevant becomes much more interesting. In this case, they come from the same IP address as the one connected to 45295/TCP. In fact, the same source has made several attempts to connect

to port 139/TCP just before starting to connect to 45295/TCP (see Figure 5). Although we don't see anything else about these two groups of flows (to 139/TCP and 45295/TCP), we're led to think that the compromise occurred right here, especially because we know that the honeypot runs a vulnerable Samba daemon on port 139. Nothing relevant for the forensics appears in the capture until 15:29:53, where an address located in the Latin American zone (LACNIC) connected to port 45295/TCP without a preliminary scan (see Figure 6). Because this port doesn't commonly have direct connections to it that aren't preceded by scans, we can reasonably suppose that the same intruder that first connected at 05:50 a.m. now controls this IP.

First traffic originating from the honeypot

Right after the connection to TCP port 45295, several flows originate from the honeypot. The first flow on TCP port 45295 targets a host in Romania on port 80. Although we can't say anything precise about the kind of traffic involved without looking at the payload, the domain name (www.pistollet.ro) combined with the destination port (80) and flow characteristic are excellent indicators that it was indeed HTTP traffic and that 534 Kbytes were downloaded. Next, two connections are made to the mail server (port 25, see Figure 7). If we had any doubts up to this point, we know now for sure that the honeypot was compromised before 15:29.

At 15:32, a machine located in a .ro domain opened a connection to port 50/TCP on the honeypot without a preliminary scan. The flow lasted 85 minutes, and many bytes traveled in two directions, indicating an interactive session such as a shell (see Figure 8).

Because this address was not seen until 15:32, and because TCP port 50 is not typical on a Linux box, we can suppose that our intruder controls this remote box and that he or

she bound some kind of shell on the honeypot's port 50/TCP. We then see several large flows, with an important number of bytes downloaded: these are, in fact, multiple attempts to connect and download tools from FTP servers. We can assume the intruder retrieved some tools from the honeypot for later use. From 16:06:35 until the end of the capture, numerous flows occur to several servers, destination port 6667/TCP; all these servers belong to the same IRC network. Those flows were concomitant in time and duration to several flows from different sources in Romania to port 31337/TCP (see Figure 9).

More shells for the kiddies!

From 16:12:49 until 23:25:46, we note several outgoing flows to various servers against several /16 networks on the Internet: nine /16s were targeted on port 139/TCP, and one /16 was targeted on port 22/TCP. At this stage of the analysis, we have learned our intruder's modus operandi and have built a timeline based on the network. The next step is to get more details from the system analysis to merge the two timelines. A follow-up article will deal with this procedure and the results obtained in more detail.

Lessons learned

Even if the flow analysis is conducted at a rather high level, we might have discovered small but sensitive traffic such as those used for covert channels. For instance, statistics on protocols are usually instructive: an HTTP session of several hours, ICMP traffic of several megabytes, or the use of an uncommon protocol should be immediately suspicious.

However, it is sometimes important to complete this analysis with one based on the packets themselves, to check that, for example, traffic is really what it looks like (such as flows to a port 443 are indeed HTTPS

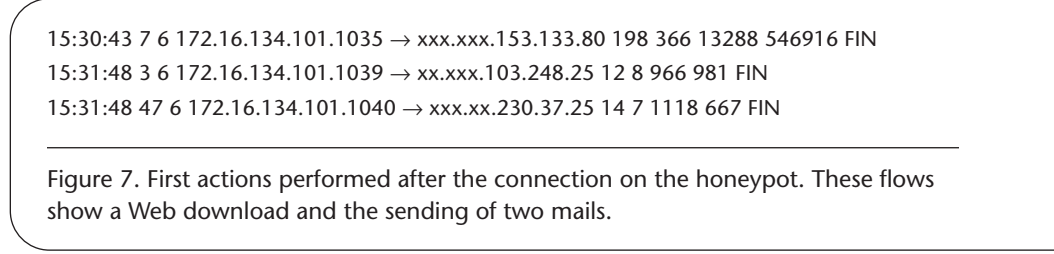


Figure 7. First actions performed after the connection on the honeypot. These flows show a Web download and the sending of two mails.

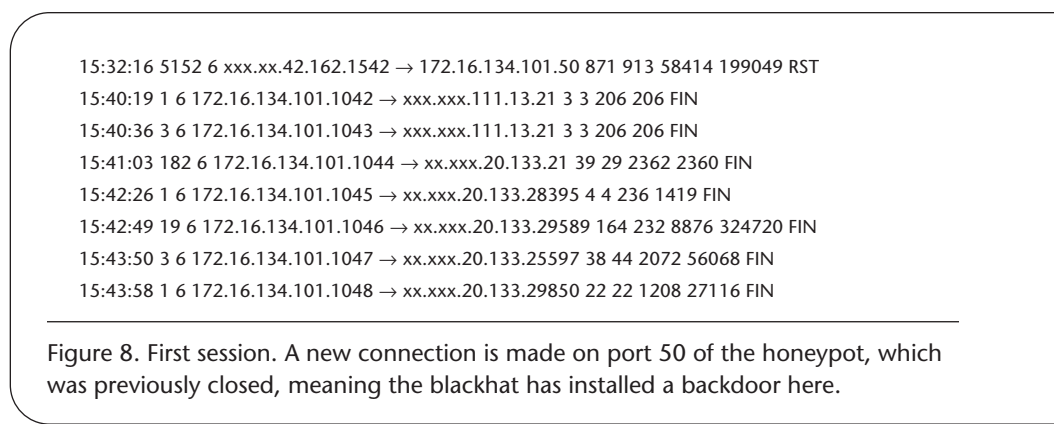


Figure 8. First session. A new connection is made on port 50 of the honeypot, which was previously closed, meaning the blackhat has installed a backdoor here.

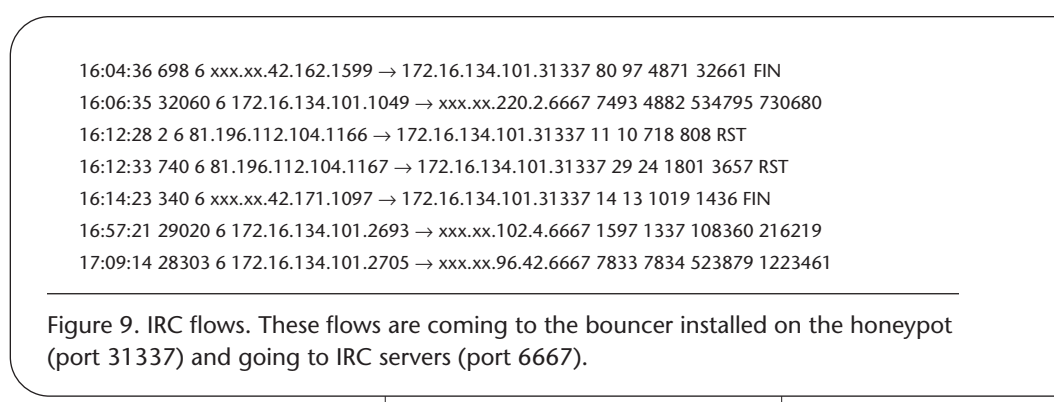


Figure 9. IRC flows. These flows are coming to the bouncer installed on the honeypot (port 31337) and going to IRC servers (port 6667).

rather than SSH traffic).

During our forensics on the honeypot, the correlation of system and network events was sufficient without going into this level of detail—system events explained the meaning of events in the network flows.

Issues with forensics and post-mortem analysis are well known. Applied to honeypots, they become larger and more complex because they're still quite a new research area. Common forensics versus the forensics applied to the honeypot differ in several ways:

- *Amount of information.* A honeypot has almost too many pieces of information, but a compromised

host doesn't have that much data.

- *Quality of information.* On honeypot, almost every piece of information is interesting if the capture tool does a good job; conversely, a compromised host usually has a lot of noise in its data.
- *Tools.* Until recently, honeypot developers focused on the honeypot itself or capture tools, but not on tools to analyze the captured data. Because a honeypot can be very talkative, investigators need specific tools to sort through all the available information. Although tools like the Sleuth Kit (www.sleuthkit.org) are useful for generic forensics, they're not good enough for honeypot forensics.

IEEE

distributed systems

ONLINE

a monthly magazine of the IEEE Computer Society

IEEE Distributed Systems Online

brings you peer-reviewed articles, detailed tutorials, expert-managed topic areas, and diverse departments covering the latest news and developments in this fast-growing field.

Log on <http://dsonline.computer.org> for **free access** to topic areas on

- **Grid Computing**
- **Mobile & Pervasive**
- **Distributed Agents**
- **Security**
- **Middleware**
- **Parallel Processing**
- **Web Systems**
- **Real Time & Embedded**
- **Dependable Systems**
- **Cluster Computing**
- **Distributed Multimedia**
- **Distributed Databases**
- **Collaborative Computing**
- **Operating Systems**

<http://dsonline.computer.org>

To receive regular updates, email
dsonline@computer.org

Our main issue was to find a way to produce good reports based on the amount of data available to us quickly. Because we had so much data, though, correlating all the pieces of information was rather difficult. A snapshot of the system would have made building the timeline easier and provided information concerning which processes were running, which files were accessed, which ports and sockets were used, and so on.

The next step in our analysis will be to look at the system side to learn what the blackhat did once he or she got in and then merge the two analyses. □

Acknowledgments

We thank Ole Stefansen, who provided the playground from which we extracted examples. We also thank David Watson (no relation to Sherlock Holmes' Watson) for the initial analysis he performed on Stefansen's honeypot. We are very grateful to the French Honeynet Project, especially Fischy (yes, you'll get your candies) for fixing our English and to Bill McCarty for his support.

Frédéric Raynal is an engineer and researcher in computer security. He also is a member of the French Honeynet Project. After three years at French Institute for Research in Computer Science (INRIA), he defended his PhD on information hiding (steganography and watermarking). He is currently the chief editor of the first French journal dealing with security, MISC (www.miscmag.com). Contact him at frederic.raynal@security-labs.org.

Yann Berthier works for HSC, a French security consulting company. He is interested in, among other things, security architecture design, intrusion detection, and forensics analysis. He is a member of the French Honeynet Project.

Philippe Biondi works as a security consultant for Arche-Omnitica Group. He is involved in pentests, R&D, and development. He has authored many open-source projects such as Scapy and LIDS. He is a member of the French Honeynet Project.

Danielle Kaminsky is a researcher and a teacher in cybercriminology. She studies behaviors related to information and information security, with her main research focus on virus writer motivations, hacker profiles, spies, and secrecy.