Global Information Assurance Certification Paper

A 2nd Generation Honeynet – Introduction, Ingredients, Setup, Deployment and Brief Results

By Kobus Jooste

0. Abstract

This report details the author's foray into the relatively new field of Honey Networks – an Information Security tool geared for research into the nature of threats.

The report starts off with an introduction to Honeypot/Honeynet technology. The document continues to describe an experimental setup deployed by the author to investigate the technology and presents how the traffic control device was set up and configured. The traffic control rules and its interaction with an inline IDS is explained fairly detailed (utilising flow diagrams) but the actual scripts implementing the configuration is not included due to their length (as is the detailed results).

Note: During the course of the author's studies (and work on this assignment) the Honeynet Project made a "Snort-Inline Toolkit" available with similar functionality as the work described in this report. Please see the "Tools" section of their webpage for further details.

1. Introduction

The term "honeynet" was coined by "The Honeynet Project" [1,2], a non-profit research organisation dedicated to learning the tools, tactics, and motives of the blackhat community and committed to share the lessons learnt in the process. The group primarily use these "honeynets" to gather their information.

Honeynets evolved from honeypots and the formal definition of a honeypot is: *A honeypot is a security resource whose value lies in being probed, attacked or compromised.* [3]

This offbeat approach may seem contrary to security measures applied to normal production resources, as these should be actively guarded from probes or attacks in any way possible.

Honeypots are flexible tools that can be applied in different situations: They can be employed to prevent attacks (through deception), similar to conventional access control devices like firewalls. They can be used to detect attacks, a task normally performed by intrusion detection systems. Another use is to capture and analyse automated attacks and or act as early indication or warning sensors.

Honeypots have no production role outside their security scope. As such no other person, application or resource should communicate with them and any activity to or from them is suspect by nature.

While honeypots can have production roles as information security measures, honeynets are a development of honeypots designed primarily for research purposes – i.e. to gather information on attackers.

They differ from production honeypots and are categorised as high-interaction research honeypots. They are not necessarily better solutions but have a different purpose - these systems gather information on threats.

A honeynet is usually a network of multiple systems subjected to probes and exploits from attackers. They utilise multiple systems at the same time like Unix and Windows systems and network hardware like routers and switches.

Honeynets recreate production networks and provide a type of laboratory where actions of would-be attackers can be carefully monitored to allow study of attackers' tools and techniques.

The technology, first employed by The Honeynet Project in 1999, continues to evolve and it is outside the scope of this assignment report to completely introduce the technology. The interested reader is referred to a white paper, [4], that gives a comprehensive explanation of the technology as well as its development.

A honeynet deployment needs to fulfil at least two critical requirements: **Data control** and **data capture**.

The first requirement prevents the honeypot from introducing an unacceptable amount of risk into the environment and the second requirement enables gainful research into the attackers' nature. Without these the honeynet merely becomes a relatively save, sponsored playing ground for attackers.

Data control is employed to mitigate the risk of a compromised system being used as a stepping-stone for further attacks against a 3rd (non-honeynet) party. It has to be implemented without compromising interactivity or raising attackers' suspicions.

The second requirement is to capture as much data around attackers' activities as captured data is essential to allow any sensible analysis of the tools and methods of attackers. Again, the challenge is to capture as much data as possible, without raising the attackers' suspicions.

The Honeynet Project has created a guideline document [5] that defines these two (and some other) requirements in detail.

Honeypot technology has currently reached its second major stage of development. The next section discusses this, more advanced (or 2nd generation) variants in more detail.

2. 2nd Generation Honeynets

The 2nd generation of honeynets evolved from fairly capable first generation technologies and the requirements can now be met with a system that is easier to deploy and yet, more difficult to detect.

Improved and more granular data control measures enables honeynet administrators to allow would-be attackers some more interactivity while, at the same time, keeping these measures relatively stealthy and difficult to detect from the attackers point of view.

The separate IDS system used for data capture and detection in previous honeynet architectures was integrated with the control device. Fulfilling the requirements with a single device, simplified deployment as all data control

and capture are handled by a single resource. The control device used in previous honeynet incarnations (an IP firewall) evolved into a layer 2 gateway device with several advantages over the conventional control device.

Layer 2 devices (like bridges and switches) are difficult to detect at the IP level (OSI Layer 3), as they do not have IP stacks. They do not route traffic and consequently there is no TTL decrement in the IP packets' headers.

These devices are generally far stealthier than conventional firewalls.

Another advantage is that all inbound and outbound traffic have to traverse the single device, making it the ideal location to control and capture data. Figure 1, below, depicts a typical layout of a second-generation honeypot network.

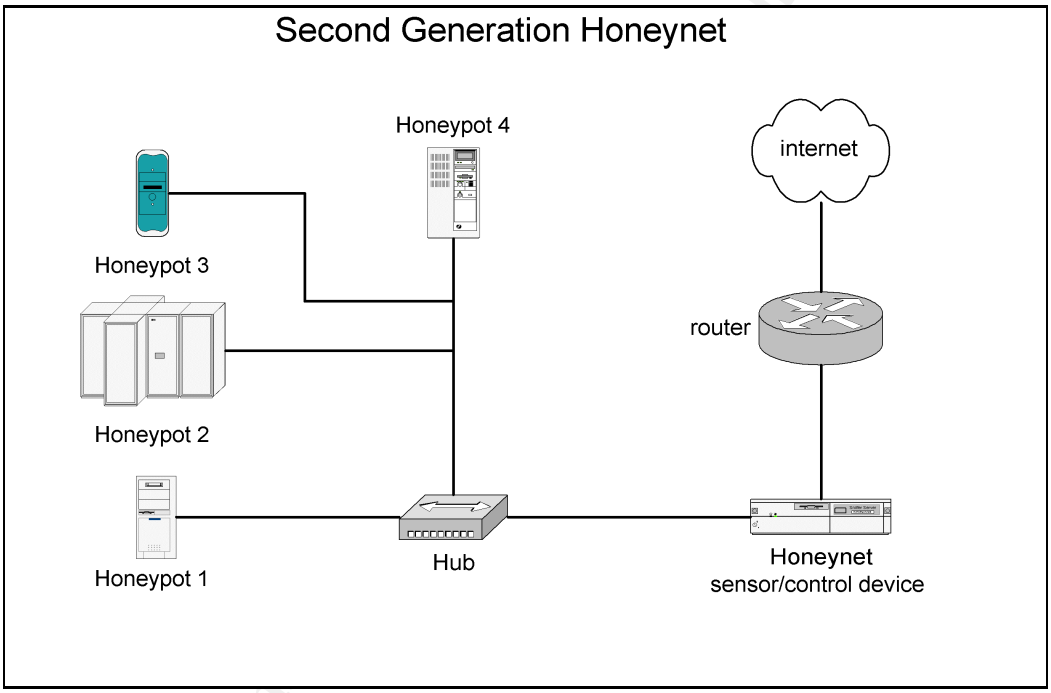

Figure 1 - Layout of a second-generation honeynet

Advanced data control enhances honeynets' abilities to mitigate risk associated with attackers' activities. Instead of merely limiting active outbound connections, more complex conditions can be specified under which activities should be allowed or disallowed. Some of these criteria allow these activities to be limited according to their possible intent.

For example, only FTP connections that exhibit signs of possible exploits can be blocked. The data control features in second-generation honeynets allow the IDS component to test the payloads of IP packets for common types of exploit codes and drop these packets.

Due to the integration of the IDS and the data control device, the IDS rules can be integrated, to a certain extent, with the packet filtering capabilities of the data control measure.

Another enhancement is that, instead of simply blocking connections, activities can be modified or throttled, rendering attacks ineffective. These enhanced measures are more difficult to detect.

Inline packet modifications further mitigate the risk without interfering unnecessarily with interesting activities. A layer 2 gateway can accomplish this by modifying the payload of suspicious packets. The attack is allowed to proceed but it will no longer be effective.

The more advanced capturing and control capabilities are implemented in two ways. The first tool is *Hogwash*. It is an IDS Gateway [6] that monitors and analyses packets and modifies or block them according to IDS signatures. Hogwash is based on Snort [7], a lightweight intrusion detection system. It installs onto a gateway system and gives that system layer 2 bridging capabilities, making it invisible to attackers and able to identify and disable attacks. The advantage of Hogwash is that it is a single software solution.

A second solution is to combine a special version of Snort recently made available, Snort-inline, with the network traffic filtering capabilities built into the Unix kernel such as iptables on Linux.

The combination produces the same functionality as Hogwash but it is slightly (in the author's opinion) more developed and stable technology at the time of writing. The disadvantage is that it requires two sets of configurations and requires two pieces of software to be running as opposed to one – iptables and snort-inline.

The honeynet built for this assignment did not use the data gathering features of the inline IDS. Instead it uses *tcpdump* software [8], an extremely flexible and capable network traffic sniffer that provides compatibility with virtually every network traffic analyser capable of reading binary dump files.

The Linux in-kernel network traffic filtering infrastructure, netfilter [9] and the bandwidth management utilities, tc, were combined with Snort-inline. A complete description of the honeynet control device is presented in the next section.

The setup utilised for the experimental honeynet uses an iptables configuration script to configure the gateway as a bridging firewall and employs the queuing mechanism of the netfilter kernel subsystem to queue appropriate traffic from kernel space into user space where the snort-inline IDS application further handles the network traffic.

The honeynet examined for this assignment employed a second layer of data control that throttles various categories' bandwidth instead of merely allowing or disallowing attacks. So instead of just blocking specific attacks, the amount of activity can be allowed but restricted

Throttling limits the capabilities for denial of service attacks or mass scanning and gives the appearance of a saturated network that increases latency. Bandwidth limitation is implemented with the in-kernel facilities and it is not documented. Refer to Project Honeynet's web pages for details of implementing bandwidth limiting under various operating systems.

3. Honeynet Configuration

Figure 2 shows the network diagram of the assignment's honeynet deployment. This section details the configuration of this particular honeynet and will proceed along the annotated numbers in the numbered diagram.

3.1 Internet Connection

The Internet connection is via a normal, medium sized business Internet connection. The main connection is via a telecomm company leased line and the backup connection is via ISDN.

The bandwidth is shared with all normal Internet traffic as the honeynet was deployed in a non-production DMZ. The bandwidth made available to honeynet traffic was only a tiny fraction on the total capacity.

3.2 Border Router

The border router is a 1600 series Cisco unit capable of handling the switchover to standby connection in case of failure.

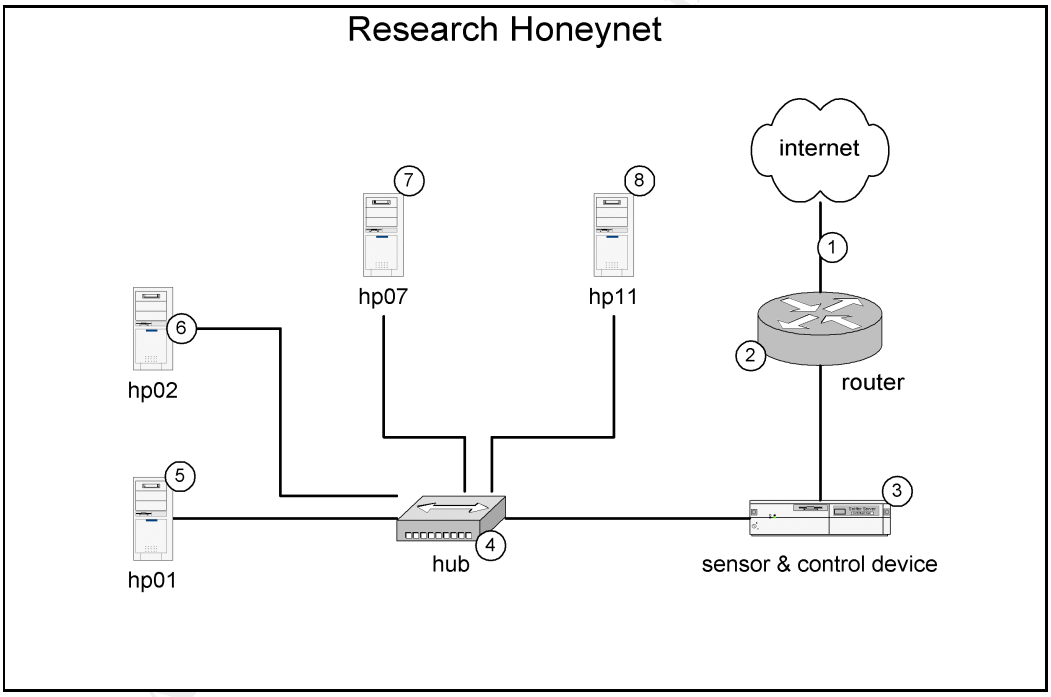

Figure 2 - Layout of research honeynet

The router was properly configured with suitable egress and ingress ACL's. Figure 3, shows the honeynet in relation to the rest of the network infrastructure. It also lists the network hardware used each segment.

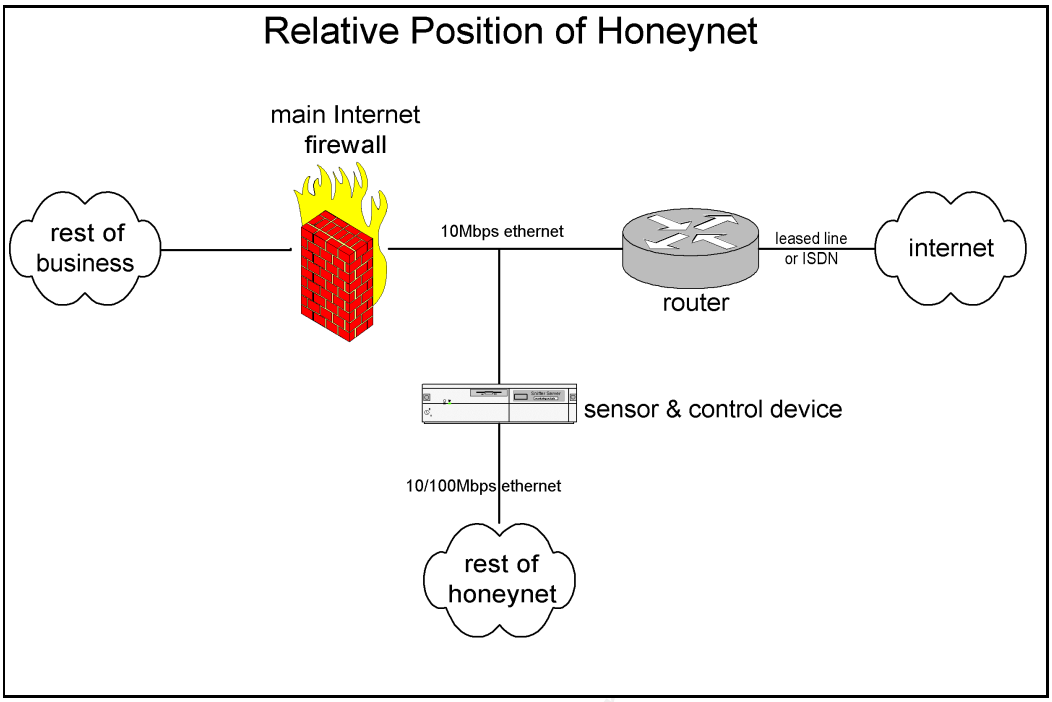

Figure 3 - Network Position of Honeynet

The diagram shows that the systems in the honeynet were basically directly exposed to the Internet. The next diagram, Figure 4, shows the IP addresses used in the honeynet setup (after sanitation) and should be useful for reference while following the descriptions ahead.

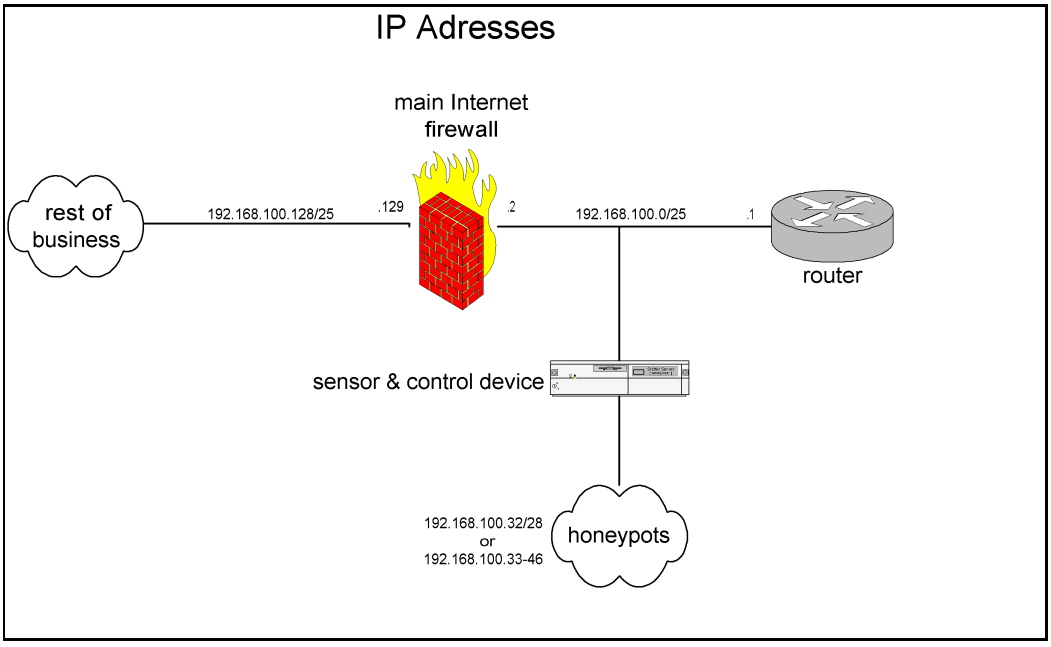

Figure 4 - Honeynet IP addresses and those in the vicinity

3.3 Honeynet Sensor and Control Device

The control device used Linux as network operating system and more specifically the Redhat 7.3 distribution of this operating system. The

distribution is an ordinary server installation with the addition of the *libpcap* and bridge-*utils packages*.

The hardware for the control device consisted of a PC with a 700MHz Intel Pentium III microprocessor, 128MB RAM and 6GB IDE hard disk drive. The computer was fitted with three ethernet network interfaces., a Realtek 8139-based card, an Intel EtherExpress Pro and a 3Com 3c905tx.

The control device used a slimmed down, custom compiled kernel but the standard Redhat kernel source package was used as it already included the ethernet bridging capabilities not found in the main tree of the 2.4 stable kernel. No kernel patches were neccesarry. In addition to this, normal responsible host security was implemented such as the disabling of unnecessary services, proper logging etc.

The recompiled kernel merely removed the subsystems and drivers that were not needed in the kernel for the control device. The essential kernel features are:

- enable development features
- enable kernel loadable modules
- choose suitable features for support of disk subsystem, character devices,other hardware and supported filesystems.
- enable networking
- enable the kernel module for each type of ethernet interface
- enable network packet filtering
- enable TCP/IP networking
- enable ethernet bridging
- enable filtering with ethernet bridging
- enable Quality of Service network feature
- enable the TBF filter under Quality of Service
- enable connection tracking for netfilter
- enable userspace queuing under netfilter
- enable netfilter's iptables support
- enable iptables' –limit match
- enable iptables' connection state matching
- enable packet filtering for netfilter
- enable the reject target for packet filtering
- enable full masquerading support under netfilter
- enable the masq and reject masquerading targets
- enable netfilter's logging target

The network configuration was enabled via a network startup script. The startup script ensured that every subsystem was correctly configured and that the necessary software and services ran at all times.

The specifics of the network configuration scripts will not be presented, but the next few paragraphs will attempt to explain the logic behind the configuration of the rules guiding the control measures. The flow of network traffic will be explained with the aid of flow diagrams.

Figure 5 shows the equivalent flow diagram for the processing that is performed for traffic arriving from the Internet and heading for one of the honeypots. The decisions made in these flow diagrams are implemented in the control device via firewall rules and the chain of events is as follows:

- A packet arrives at the external (Internet facing) interface. The packet is received from the Internet and headed toward one of the honey pot systems behind the control device.
- The control device acts as a layer 2 bridging device so it will intercept the packet even tough it is not involved in the IP exchange between the two hosts. The control device accepts the packet on behalf of the destination honeypot.
- The external ethernet interface, denoted with eth2 in the firewall rules, does not have an IP address. It is invisible to both the sender of the packet and the receiving honeypot.
- After the packet is intercepted it is subject to decisions and processing steps that determine whether it will eventually be forwarded or not.
- The first decision made by the control device is whether the source address is valid or not; if not the packet is dropped and processing ceases immediately.

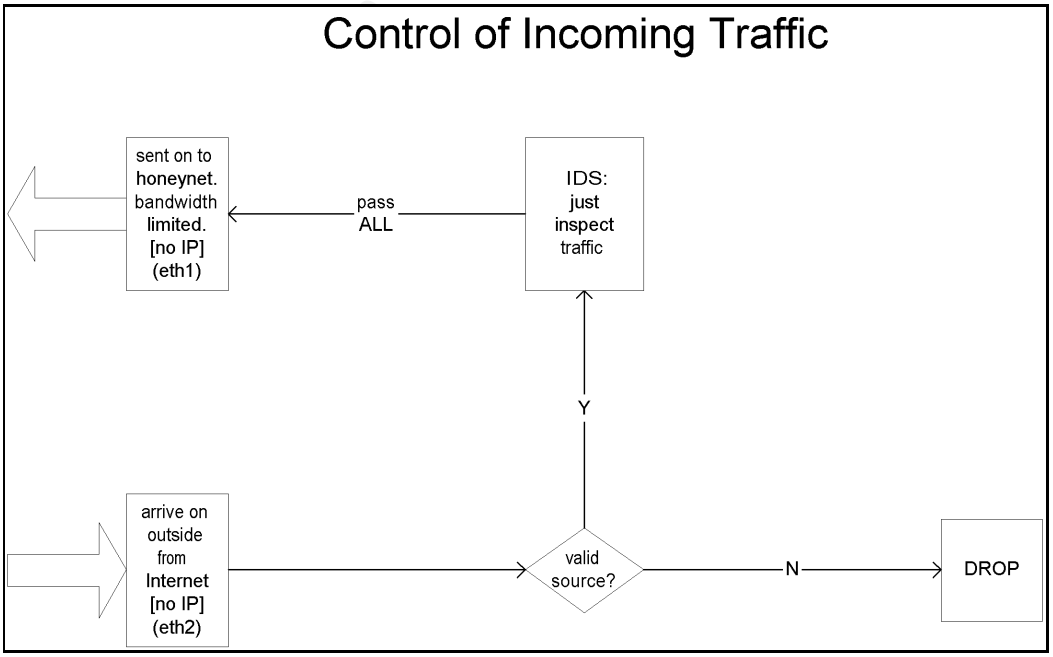

Figure 5 - Processing of Incoming Honeynet Traffic

- Valid source addresses are any addresses except the border router (192.168.100.1) and firewall (192.168.100.2), as we do not want these sensitive resources probed by a compromised honeypot.
- If valid, the packet is passed along to the IDS where it will be subject to inspection to see if it matches any signatures contained in the IDS's database. Regardless of whether it matches or not, the IDS will pass the packet along for further processing.
- At last the packet is passed on to the inside interface, eth1, on the honeypot network's side from where it will reach its destination. The interface performs bandwidth limiting as a final step before the packet is put onto the network cable.

Figure 6, further on, shows the equivalent flow diagram for the processing that is performed for traffic destined for the Internet and originating from one of the honeypots. The chain of events is as follows:

- A packet arrives at the internal (honeypot facing) interface. The packet is headed towards the Internet and originated from one of the honeypot systems behind the control device.
- The control device acts as a layer 2 bridging device so it will intercept the packet even tough it is not involved in the IP exchange between the two hosts. The control device accepts the packet on behalf of the destination system.
- The internal ethernet interface, denoted with eth1, in the firewall rules, does not have an IP address. It is invisible to both the source and destination hosts.
- After the packet is intercepted it is subject to a number of decisions and processing steps that determine whether it will eventually be forwarded or not.
- The first decision made by the control device is whether the source address is valid or not, and, if not the packet is dropped and processing ceases immediately.

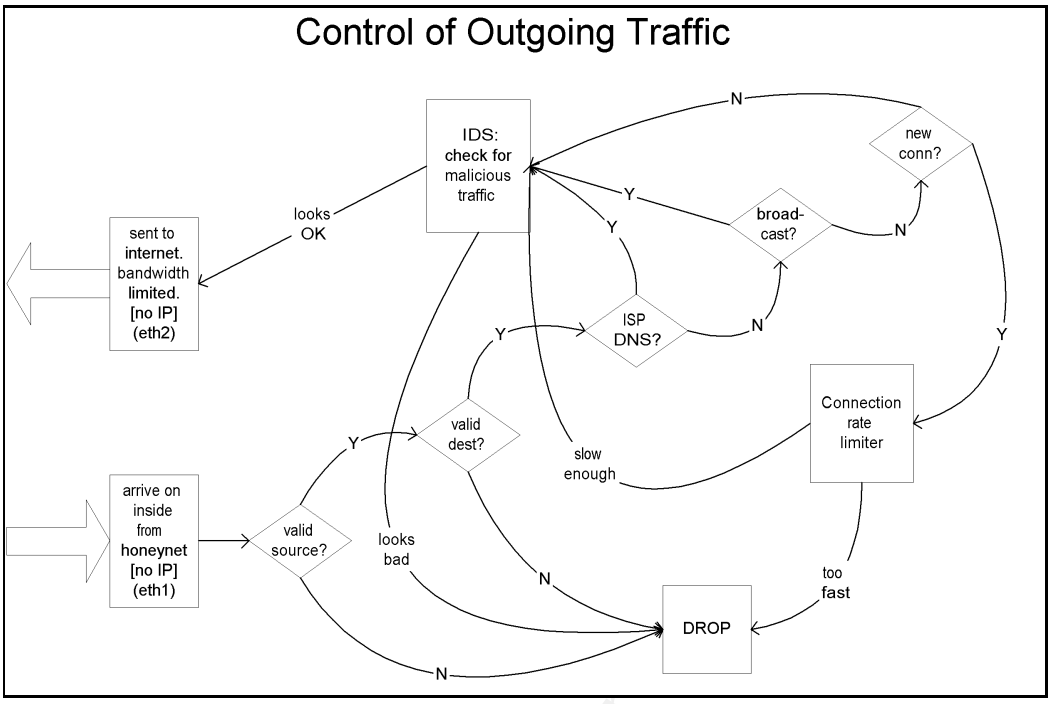

Figure 6 - Processing of Outgoing Honeynet Traffic

- Valid source addresses are any addresses in the range of the honeypot systems, i.e. 192.168.100.33-46. Source addresses outside this range indicate that something out of the ordinary happened, either a network failure or misconfiguration but more likely it means that an attacker is using a compromised honeypot and faking a source address while trying to communicate with the outside world.
- The next decision made by the control device is whether the destination address is valid or not. If not the packet is dropped and processing ceases immediately.
- Valid destination addresses are any addresses except the border router (192.168.100.1) and firewall (192.168.100.2) as we do not want these sensitive resources probed by a compromised honeypot. If valid the packet is passed on.
- The next decision made by the control device is whether the traffic is a DNS related query to the ISP's name server, if so the packet is queued to the IDS (see latter points for IDS processing) if not he packet is passed along for further processing.
- The next decision made by the control device is whether the traffic is broadcast traffic, if so the packet is queued to the IDS (see latter points for IDS processing) if not he packet is passed along for further processing.
- The next decision made by the control device is whether the packet initiates a new connection, if not if means that the packet is part of an already established connection that was allowed at a previous stage so the packet is sent on to IDS processing. (see latter points for IDS processing)

- If a new connection is attempted the packet is passed on to the connection rate limiting logic.
- The connection rate limiting logic determines whether the new connection is under or above the threshold that limits the rate and number of new connections. If the connection exceeds the rate limitations the packet is dropped and processing ceases, but if the connection is within limits it is passed on to the IDS processing step.
- IDS examines packets passed along to it and compares it to known exploits stored in the IDS's databases. If any matches are encountered the packet is dropped and processing ceases but if the IDS regards the packet as not malicious, the packet is passed along once more.
- At last the packet is passed on to the outside interface, eth2, on the Internet from where it will be routed, first by the border router and them all they way to its destination on the Internet. The interface performs bandwidth limiting as a final step before the packet is put onto the network cable.

The complete statement-by-statement description of the network configuration script won't be presented but it can be summarised in the following steps:

- Load kernel modules
- Ensure IDS is running
- Initialise network interfaces
- Enable and configure bandwidth limiting
- Configure ethernet bridging
- Flush stale firewall rules
- Install suitable firewall rules (the description via the flow diagrams gave a good overview of these rules)
- Initialise the network traffic capturing software
- Initialise the bridge

3.4 Honeypot network Hub

This particular honeynet used a 10/100Mbps ethernet switch to connect the honeypots to the network and to the control device. The control device was connected to a monitoring port on the switch so that it would be able to observe and capture all traffic on the honeypot network. So from the control device's perspective if was indeed a hub but the other systems operated in a normal switched environment.

3.5 Honeypot 1, hp01, 192.160.100.34

Operating System:

Microsoft Windows 2000 SP2

Configuration:

- Standard IP services running

- Default IIS webserver running
- Default MS FTP service running
- Default SMTP service running
- Configured as Microsoft client
- No filesharing configured
- Disabled NetBios over TCP/IP

3.6 Honeypot 2, hp02, 192.160.100.35

Operating System:

Microsoft Windows 2000 SP2

Configuration:

- snmp service running
- message queuing enabled
- rip (routing) service enable
- Unix print services enabled
- Configured as Microsoft client
- Configured with Microsoft filesharing
- Configured with NetBios over TCP/IP

3.7 Honeypot 3, hp07, 192.160.100.40

Operating System:

Sun Solaris 8 on Sun Ultra 5 workstation

Configuration:

- Default install
- DNS service enabled
- SSH service enabled

3.8 Honeypot 4, hp11, 192.160.100.44

Operating System:

Linux, Redhat 7.2

Configuration:

- Default Installation

4. Some Results

This section very briefly mentions some of the interesting detects captured by the inline intrusion detection system employed on the control device. Even brief discussions supported by minimal packet traces are too voluminous to include in this report. Traffic is divided into subsections according to their IP

protocol. The capitalised words, below, indicates the category of the matching rule in the IDS rule base.

4.1 ICMP

ICMP unreachable messages elicited by portscans were filtered out of the total number of ICMP detects as these messages are not attacker generated traffic but is generated according to the Internet Control Message Protocol mostly in response to portscans directed to non-listening hosts and services.

The remaining ICMP traffic is about a total of 1000 ICMP Ping (code 8, type 0) and ICMP Echo Reply (code 8, type 0) packets in roughly a 1:1 ratio. These messages emanated from roughly 110 different source addresses. Giving and average of roughly 5 pings sent per host.

4.2 UDP

The majority of traffic is RPC related confirming RPC's place in the "top 20 most vulnerable or prone to attack services" [10] list.

RPC attacks:

- RPC portmap request status
- RPC portmap request sadmind
- RPC EXPLOIT statdx
- EXPLOIT ntpdx overflow attempt

Note: None of these attacks were successful

4.3 TCP

The list below present some interesting TCP traffic captured by the inline IDS. The majority of traffic was proxy scanning (not presented here) with web related (MS and UNIX) attacks in 2nd place. Here are some of the web related detects:

- WEB-IIS ISAPI .ida access
- WEB-MISC /etc/passwd

Another interesting detect was not specifically attack generated but nevertheless attempted to misuse some internet service:

- POLICY SMTP relaying denied

The total number of detects are too numerous to present in this report, even in a digested form.

Note: None of these detected attacks were successful. The author chose fairly conservative honeypot configurations.

References

1. Honeynet Project, "HoneyNet Prioject" URL: http://project.honeynet.org/

2 The Honeynet Project, Know Your Enemy: Revealing the security tools, tactics, and motives of the Blackhat community, Addison Wesley 2002, p 9-17

3 Spitzner, Lance, Honeypots, Tracking Hackers, Addison Wesley 2003, p 40

4. Honeynet Project , "Know Your Enemy: Honeynets" 7 Jan 2003 URL: http://project.honeynet.org/papers/honeynet/

5. The Honeynet Alliance, " Honeynet Definitions, Requirements, and Standards" Version 1.5.0, 7 Jan 2003 URL: http://project.honeynet.org/papers/alliance/requirements/

6. Larsen, J., "Hogwash" URL: http://hogwash.sourceforge.net/

7. Roesch, M. & Caswell B. "snort.org" URL: http://ww.snort.org/

8. "TCPDUMP public repository" 12 Dec 2002 URL: http://www.tcpdump.org/

9 Frost, S. & The Netfilter Web Team "netfilter/iptables" URL: http://netfilter.samba.org/

10. The SANS Institute, "SANS / FBI The Twenty Most Critical Internet Security Vulnerabilities" Version 3.21 17 Oct 2002 URL: http://www.sans.org/top20/

Last Updated: May 28th, 2017

Upcoming Training

SANS Stockholm 2017	Stockholm, Sweden	May 29, 2017 - Jun 03, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
Security Operations Center Summit & Training	Washington, DC	Jun 05, 2017 - Jun 12, 2017	Live Event
Community SANS Ottawa SEC401	Ottawa, ON	Jun 05, 2017 - Jun 10, 2017	Community SANS
SANS San Francisco Summer 2017	San Francisco, CA	Jun 05, 2017 - Jun 10, 2017	Live Event
SANS Charlotte 2017	Charlotte, NC	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Rocky Mountain 2017 - SEC401: Security Essentials Bootcamp Style	Denver, CO	Jun 12, 2017 - Jun 17, 2017	vLive
SANS Secure Europe 2017	Amsterdam, Netherlands	Jun 12, 2017 - Jun 20, 2017	Live Event
Community SANS Portland SEC401	Portland, OR	Jun 12, 2017 - Jun 17, 2017	Community SANS
SANS Rocky Mountain 2017	Denver, CO	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Minneapolis 2017	Minneapolis, MN	Jun 19, 2017 - Jun 24, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Cyber Defence Canberra 2017	Canberra, Australia	Jun 26, 2017 - Jul 08, 2017	Live Event
SANS Paris 2017	Paris, France	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, Japan	Jul 05, 2017 - Jul 15, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Minneapolis SEC401	Minneapolis, MN	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Los Angeles - Long Beach 2017	Long Beach, CA	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Phoenix SEC401	Phoenix, AZ	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Munich Summer 2017	Munich, Germany	Jul 10, 2017 - Jul 15, 2017	Live Event
Mentor Session - SEC401	Ventura, CA	Jul 12, 2017 - Sep 13, 2017	Mentor
Mentor Session - SEC401	Macon, GA	Jul 12, 2017 - Aug 23, 2017	Mentor
Community SANS Colorado Springs SEC401	Colorado Springs, CO	Jul 17, 2017 - Jul 22, 2017	Community SANS
Community SANS Atlanta SEC401	Atlanta, GA	Jul 17, 2017 - Jul 22, 2017	Community SANS
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
SANSFIRE 2017 - SEC401: Security Essentials Bootcamp Style	Washington, DC	Jul 24, 2017 - Jul 29, 2017	vLive
Community SANS Charleston SEC401	Charleston, SC	Jul 24, 2017 - Jul 29, 2017	Community SANS
Community SANS Fort Lauderdale SEC401	Fort Lauderdale, FL	Jul 31, 2017 - Aug 05, 2017	Community SANS
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event