

# Honeypot Farms Using Ethernet Bridging over a TCP Connection

Bruno Morisson <bruno.morisson@honeynet-pt.org>

Marco Vaz <marco.vaz@honeynet-pt.org>

Pedro Inácio <Pedro.inacio@honeynet-pt.org>

*The Honeynet Project - Portugal*

<http://www.honeynet-pt.org>

*March 2005*

## **Abstract**

*This paper describes a methodology and a prototype tool based on libpcap, libnet, mini-lzo and openssl, with the main objective of mitigating honeypot farm deployment complexity. Ethernet bridging over a TCP connection and vlan tagging are used as the way to simplify the deployment of the remote honeypots and achieve better overall results.*

## **Keywords**

*Honeypot, honeynet, ethernet bridging, vlan, snort-inline, TCP*

## Introduction

As already presented by Edward Balas in chapter 7 of “Know Your Enemy, 2nd Edition”, Honeypot Farms are used as a way of virtually distributing honeypots, transporting IP packets from remote locations to the physical honeypots. It aims to reduce cost, deployment time and analysis time.

We decided to embrace and simplify this concept of traffic tunneling because it gives us the ability to optimize our methodology of analysis. Our aim was to use bridging rather than routing for transporting the traffic from remote locations to our honeypots on the farm. Bridging provides the simplicity of traffic transportation.

This approach enables:

- Quickly access to the compromised equipments.
- Reduce the physical space required in ISP housing.
- Decrease the number of control equipments (bridge, firewall and database)
- Transparency

As Edward Balas puts it:

Advantages of honeypot farms

- Honeynets can be deployed with in a very short amount of time.
- Forensic analysis can be done faster.
- Honeypot farms can be used to protect production servers (hot-zoning).
- Participant networks don't need to configure or monitor the honeypots.

Disadvantages of honeypot farms

- Geographic unrelated positions cause anomalies in network latency.

- Honeypot farms use routing rather than bridge, so they are complex to configure and require good network knowledge to operate properly.
- This technology is fair new, there are no tools to help automate the configuration and operation of the infrastructures.

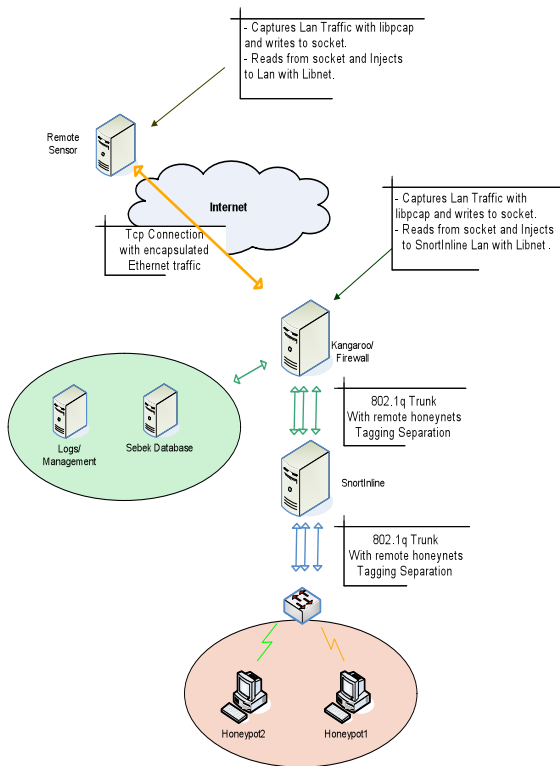
The Portugal Honeynet Project developed its own prototype tool, Kangaroo, to help us with those tasks. With this tool, it is possible to tunnel ethernet traffic from remote locations to the farm on a TCP connection, injecting it in the physical honeypot Ethernet card/switch.

Since Kangaroo relies on *libpcap* and *libnet* for capturing and injecting packets, it doesn't require any kernel patches/modules, and it should be simple to port it to most modern operating systems.

## Architecture

The topology deployed is based on a Gen II honeynet. The big difference is that we are isolating the honeypots using 802.1q, creating virtual interfaces on our bridge server. The bridge server has one virtual interface per network segment.

All traffic to and from the honeypots will be bridged by the central server and the remote servers. The remote servers will create TCP connections to the central server, through which interesting traffic will be tunneled both ways.



#### Traffic to the honeypot:

The remote client captures the Ethernet frames sent to the honeypot's IP address (as well as relevant ARP requests/answers), and sends the frames through the TCP connection to the central server. The central server receives the frames from the TCP connection, and injects them in the desired interface.

#### Traffic from the honeypot:

The central server captures the Ethernet frames sent from the honeypot's IP address (as well as relevant ARP requests/answers), and sends the frames through the TCP connection to the remote client. The remote client receives the frames from the TCP connection, and injects them on the local network.

#### BPF - BSD Packet Filter

Using *libpcap* to capture the packets gives the possibility to create a set of powerful filtering rules that will reduce the transported traffic to the minimal

operational needs. Filtering rules can also be applied via BPF on both ends of the tunnel.

Another important advantage is that *libpcap* makes Kangaroo a truly platform-independent application.

#### Transparency

This approach to traffic tunneling can be described as medium-independent and inherently protocol-independent. Any LAN medium that can be captured with *libpcap* and injected with *libnet* can be tunneled with Kangaroo. Contrary to routing, using bridging there is no way for an attacker to detect that the machine is not physically located next to the others on the same subnet, except for the added latency.

#### Performance

Latency: There is, obviously, added latency to the connection. To reduce it, Kangaroo offers the ability to compress the packets.

Scalability: For now, one instance of Kangaroo is required for each tunnel created, so for now scalability is an issue at that level. It is also an issue with the bridge, since there is only one snort inline instance which must process every packet for every honeypot.

#### Example Setup

In the example setup there are two remote sensors with Kangaroo that will forward and receive the traffic to and from the honeynet Firewall.

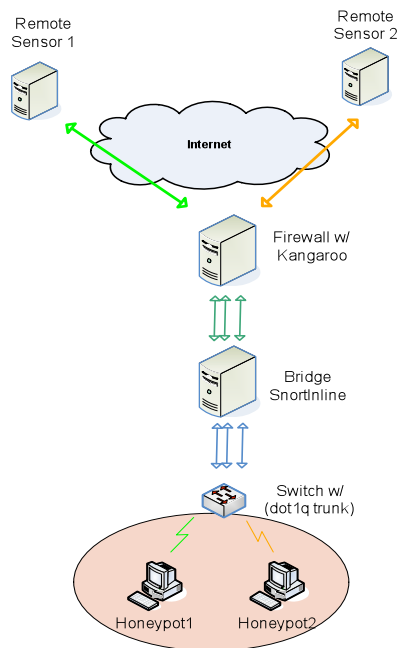
The firewall will encapsulate each stream into dot1q and send the trunk back to the Bridge.

The SnortInline bridge has several virtual bridges and each one is composed by pairs of dot1q interfaces. One SnortInline process is enough to process all bridges.

The switch located behind the SnortInline Bridge will split the trunk traffic, putting each VLAN in a separate port.

There are two honeypots, one on VLAN 2 and other on VLAN 3.

Honeypot1 will have the (public) IP address 99.x.y.z and honeypot2 will have the (also public) IP address 100.x.y.z. The IP address for the firewall's outside interface (public) will be 80.x.y.z



#### Configuration on the remote sensors

- `./kangaroo -C -u user -g group -p 4791 -c 80.x.y.z -i eth0 -f 'dst host 99.x.y.z'`  
Kangaroo will capture and inject traffic from and to RemoteSensor1
- `./kangaroo -C -u user -g group -p 4792 -c 80.x.y.z -i eth0 -f 'dst host 100.x.y.z'`  
Kangaroo will capture and inject traffic from and to RemoteSensor2

#### Configuration on the Firewall

- `vconfig add eth1 2` Creation of the VLAN 2 interface on the inside Ethernet NIC.
- `vconfig add eth1 3` Creation of the VLAN 3 interface on the inside Ethernet NIC.

- `./kangaroo -C -u user -g group -p 4791 -l 80.x.y.z -i eth1.2 -f 'src host 100.x.y.z'`  
Kangaroo will inject traffic coming from RemoteSensor to the VLAN 2 interface on the firewall
- `./kangaroo -C -u user -g group -p 4792 -l 80.x.y.z -i eth1.3 -f 'src host 99.x.y.z'`  
Kangaroo will inject traffic coming from RemoteSensor to the VLAN 3 interface on the firewall

#### Configuration on the SnortInline Bridge

- `vconfig add eth0 2` Creation of the VLAN 2 interface on the inside Ethernet NIC.
- `vconfig add eth1 2` Creation of the VLAN 2 interface on the outside Ethernet NIC.
- `vconfig add eth0 3` Creation of the VLAN 3 interface on the inside Ethernet NIC.
- `vconfig add eth1 3` Creation of the VLAN 3 interface on the outside Ethernet NIC.
- `brctl add eth0.2 br2` Creation of the VLAN 2 interface on the inside Ethernet NIC.
- `brctl add eth1.2 br2` Creation of the VLAN 2 interface on the outside Ethernet NIC.
- `brctl add eth0.3 br3` Creation of the VLAN 3 interface on the inside Ethernet NIC.
- `brctl add eth1.3 br3` Creation of the VLAN 3 interface on the outside Ethernet NIC.
- `iptables -A FORWARD -s 99.x.y.z -j ACCEPT` Firewall rule to permit inbound traffic to the honeypot1.
- `iptables -A FORWARD -d 99.x.y.z -j QUEUE` Firewall rule to queue outbound traffic from the honeypot1.

- `iptables -A FORWARD -s 100.x.y.z -j ACCEPT` Firewall rule to permit inbound traffic to the honeypot2.
- `iptables -A FORWARD -d 100.x.y.z -j QUEUE` Firewall rule to queue outbound traffic from the honeypot2.

## TODO

- The prototype code is being completely re-written.
- Multiplexing - One central Kangaroo daemon for several remote servers.
- Direct integration with snort-inline
- Performance enhancements
- SSL enhancements (Certificates for authentication, stream encryption for performance)

## Conclusions

Using the methods described, with the aid of Kangaroo, Linux bridging, VLAN tagging, and snort inline, it is possible to truly decentralize honeynets, deploying honeypots remotely, while keeping all the hardware centralized at a close location. This also allows for costs reduction on hardware: there is no need to replicate honeynets in every participating network. We can use the same firewall, bridge, log server, etc., for every honeypot deployed. With Kangaroo routing configuration troubles can be avoided, and network transparency obtained. Since Kangaroo was specifically thought for this application, its configuration is also simple and straightforward.

## References

Kangaroo  
<http://www.honeynet-pt.org/research/kangaroo-0.5.0a.tar.gz>

Libpcap  
<http://www.tcpdump.org>

Libnet  
<http://www.packetfactory.net>

Mini-LZO  
<http://www.oberhumer.com>

Openssl  
<http://www.openssl.org>

The HoneyNet Project  
<http://www.honeynet.org>

Book - Know Your Enemy, 2<sup>nd</sup> Edition  
<http://www.honeynet.org/book>