SURFnet Cyber Intelligence Framework
HoneyPot design document
v0.1

Gijs Molenaar

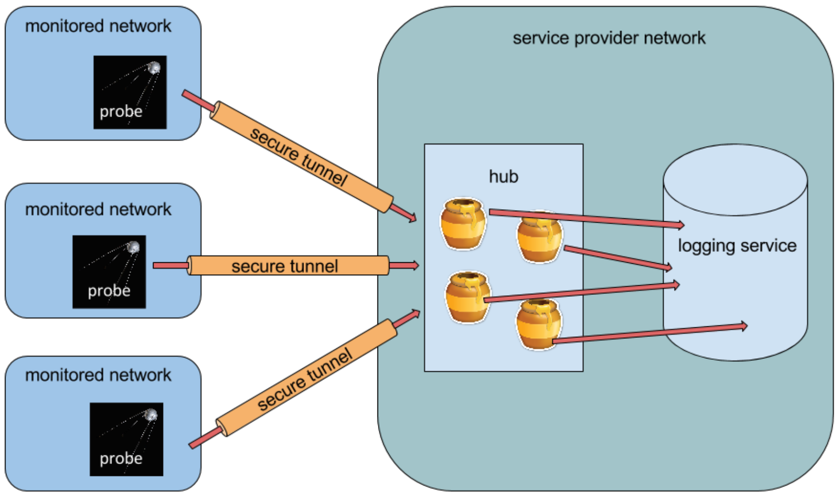
1. Introduction

Quick introduction about what are we making, why are we making this and who is the target user.

2. Design

2.1. Key components / terminology

- **Honeypot service provider network** - the entity that hosts the attack detection software.
- **Monitored network** - the network which is beeing monitored for incomming attacks.
- **Secure tunnel** - medium for transporting traffic from monitored network to service provider network.
- **logging service** - Log collection & analysis tool/service. Typically running at service provider.
- **Hub** - running the actual honeypot(s). Typically running at service provider.
- **Probe** - zero config entity that 'calls home' and routes incoming (attach) traffic to the hub. Typically running in monitored network.



2.2. The network

In the simplest case all key components described above are running on the same machine and the service provider network is equal to the monitored network. This will probably happen during the initial phases of development. In practice this wont be the case.

2.3. The probe

We want to keep the probe as simple as possible. Zero or nihil configuration, zero mainteance. Idea's are bootable USB sticks but also simple embedded systems. Still it should be possible to easily install the software on a 'normal' (desktop) computer.

2.4. The logging service

For now all eyes are on kibana & logstash by elastic search. Logstash for storing logfiles into Elasticsearch. Kibana is an interactive data visualization and exploration tool.

3. Technical decisions made

Where these is a choice of programing language we will program in Python (2.7/3.x compatible).

We want to package up all software and set up a software distribution platform.

We will focus on red hat enterprise linux 7 compatibility, which is equal to CentOS 7 compatibility.

SURFnet is providing us with various virtual machines for testing and deployment.

Development is centralized on github. The project name is cyberintelframework. The old SURFids software has been moved from SVN to this git repository also. Entrypoint repo is the 'main' reposiry, where also the documentation and notes are stored.

4. Technical open questions

idea is to progressivly find an answer to these questions and move them to the 'decisions made':

- How to transfer the network traffic from the probe to the hub? Use honetrap-agent or VPN?
- Make it possible to run Hub in proximity of probe? No monitored network traffic leaves the network, possibly less traffic?
- What off-the-self honeypot system are we going to use?
- honeypot management. How do we start/stop/control the honeypots? We may use honeytrap for that. Or we use honeytrap as one of the honeypots for high interaction?
- are the old SURFids scripts of any use?
- What are we going to store?
- Does CentOS properly support USB live? maybe
- Encapsulating the honeypots. Is it useful to use containerization for honeypots? In the case of high interaction honey pots yes, since the whole concept is based on this. But is it useful for low interaction honeypots or is it just a obstruction?
- Is docker a good container solution?

5. links

5.1. similar products

- T-pot
- honeytrap & honeycast
- SURFids (depricated)

5.2. Honeypots and instruction detection software

- bro
- kippo
- glastopf

5.3. SURFids code

- tunnel
- sensor
- packaging-logserver
- packaging-tunnel
- logserver
- argos

5.4. VPN related

- eduVPN

5.5. others

- debops