Research Project 2: Metasploit-able Honeypots

Wouter Katz
wouter.katz@os3.nl

University of Amsterdam

July 4th 2013

Research questions

How feasible is an automated method to detect specific exploits on a honeypot by monitoring network traffic of exploits?

- What setup is needed in order to have exploits successfully complete their exploit against a honeypot?
- What is the best method to process network traffic to/from the honeypot to extract and match a unique signature from exploit traffic?
- How successful are these methods?

Research questions summarized

Protocol independent

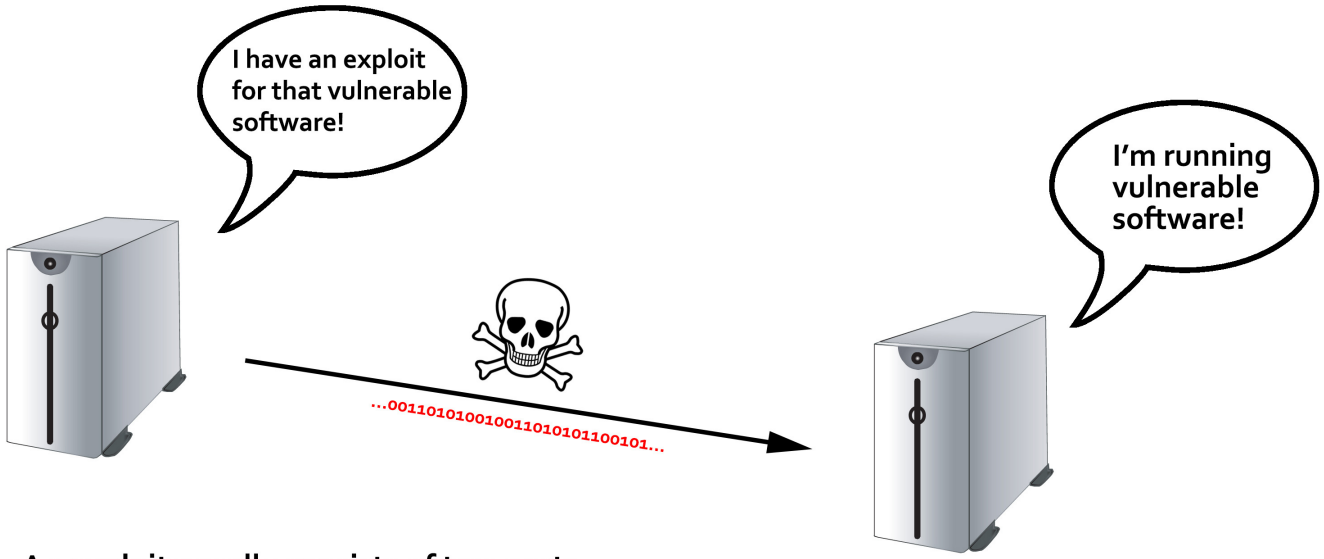
Introduction

"A honeypot is [...] a resource which is intended to be attacked and compromised to gain more information about the attacker and the used tools." (Baumann & Plattner, 2002)

Introduction

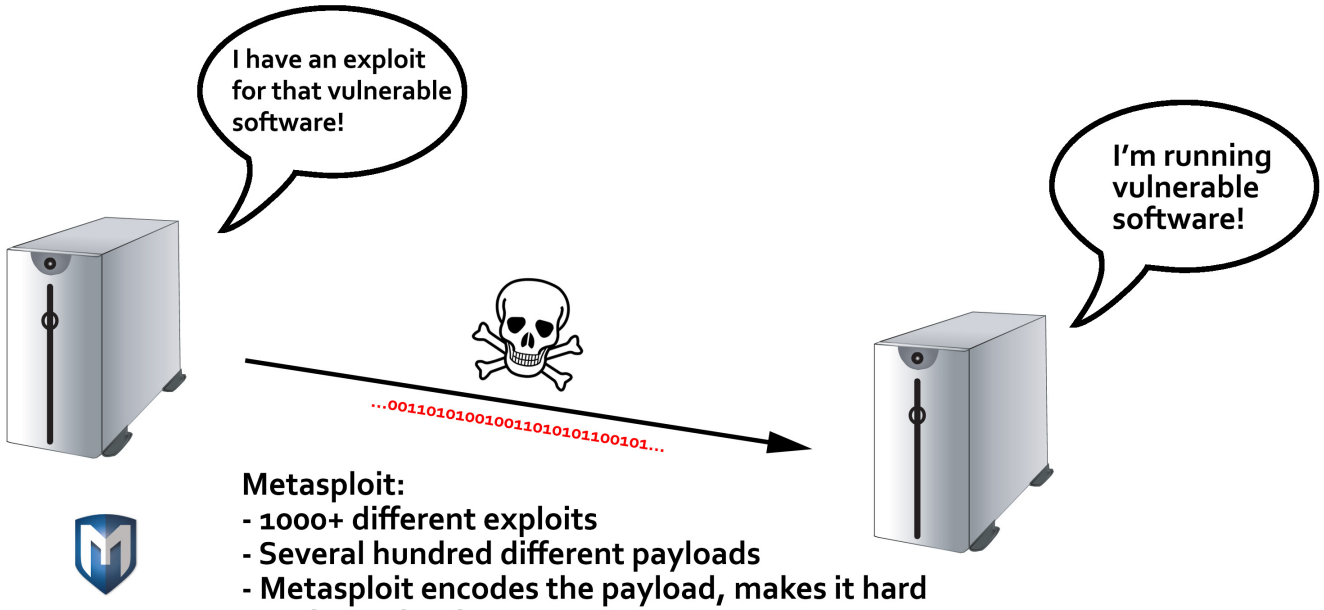
An exploit is used to abuse a security vulnerability, leading to an attacker gaining unintended privileges.
(Anley et al., 2011)

Introduction



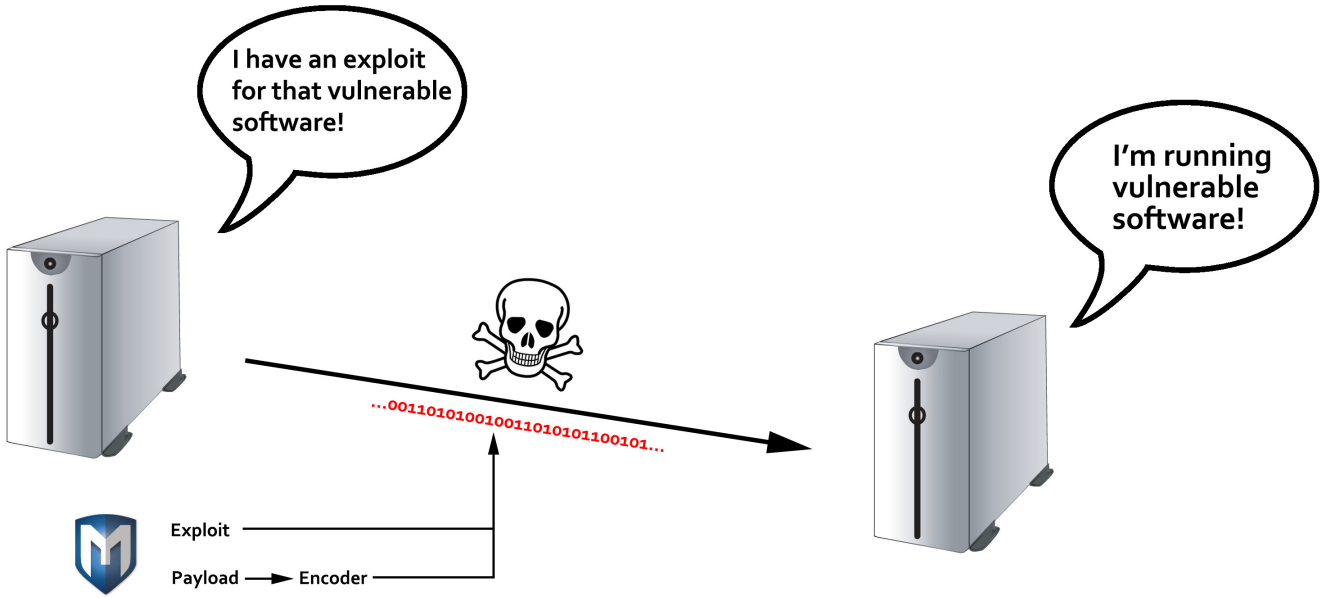
An exploit usually consists of two parts:
- First trigger the vulnerable application to execute custom code
- The "payload", containing the code to be executed

Introduction

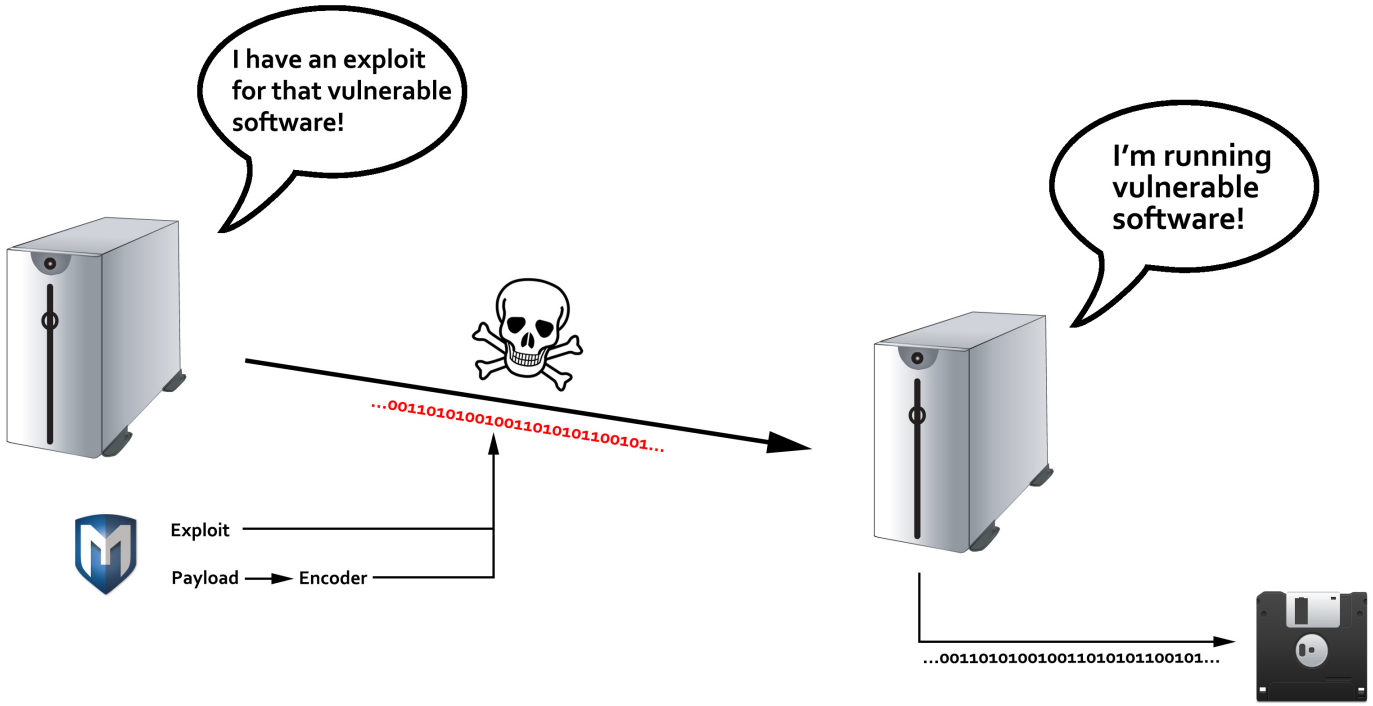


Metasploit:
- 1000+ different exploits
- Several hundred different payloads
- Metasploit encodes the payload, makes it hard to detect by signature
- Easy to use: choose an exploit, choose a payload to include, fire away!

Introduction



Introduction



Why is this needed?

- A lot of the honeypot software contain outdated vulnerabilities
- Analysis of what happened requires manual analysis
- Having signatures for the most-used penetration testing tool allows for valuable insight in attackers' activities

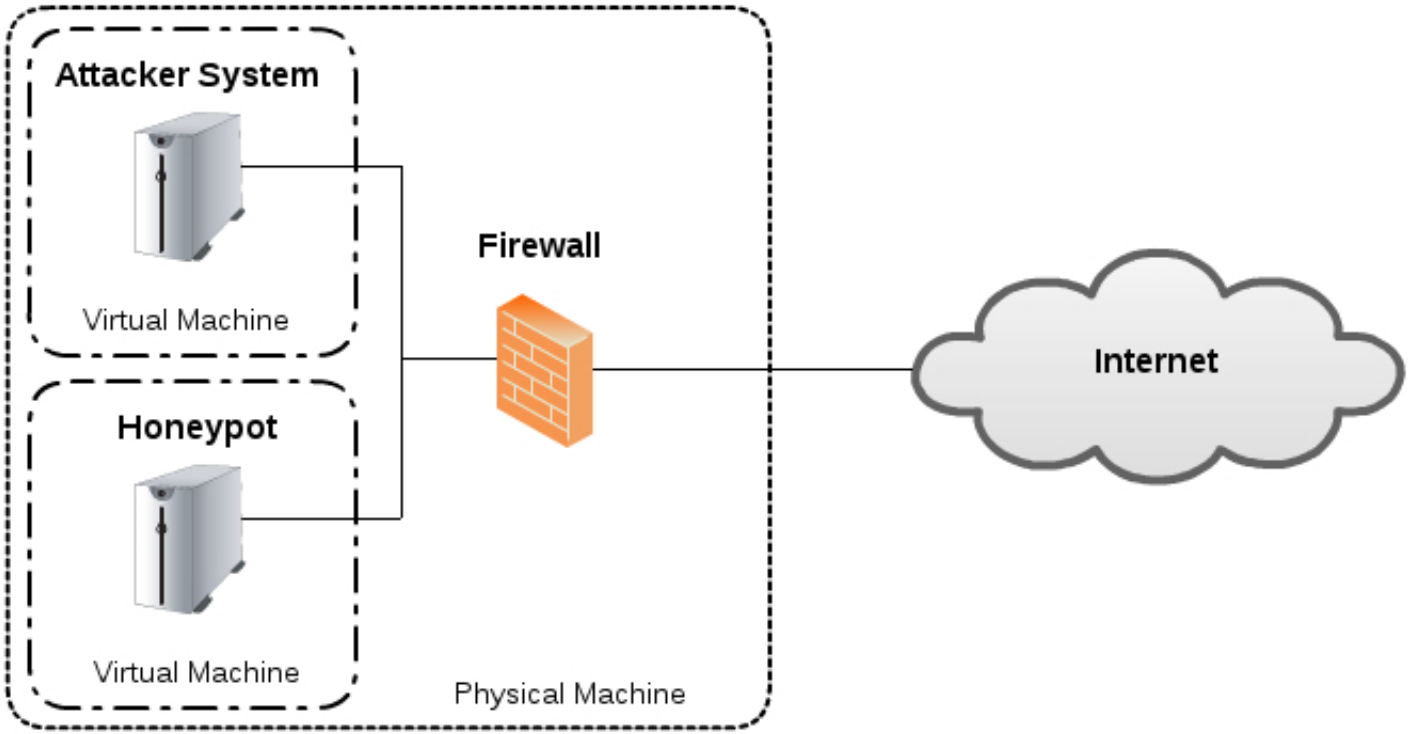
What we want is to automatically detect modern exploits and show which exploits were detected.

Exploits used within Metasploit

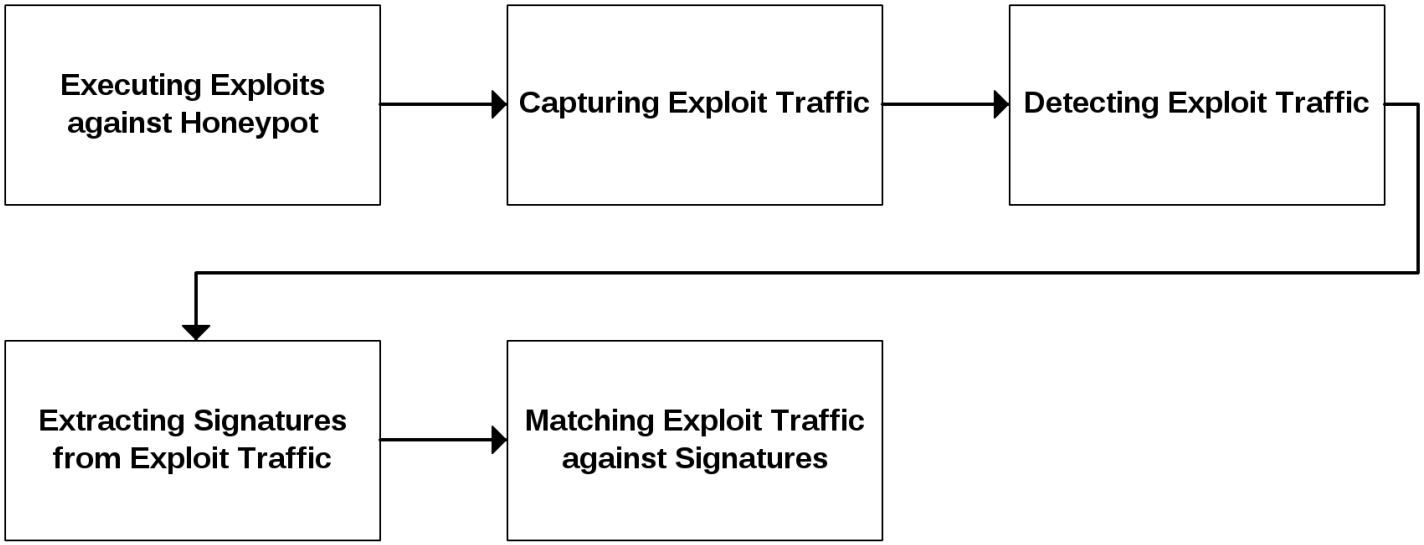
Within Metasploit, exploits targeting FTP server software were chosen as a test set for the research:

- Large number of exploits (37)
- FTP is plain-text protocol, makes development easier
- Simple commands/responses

Testing environment



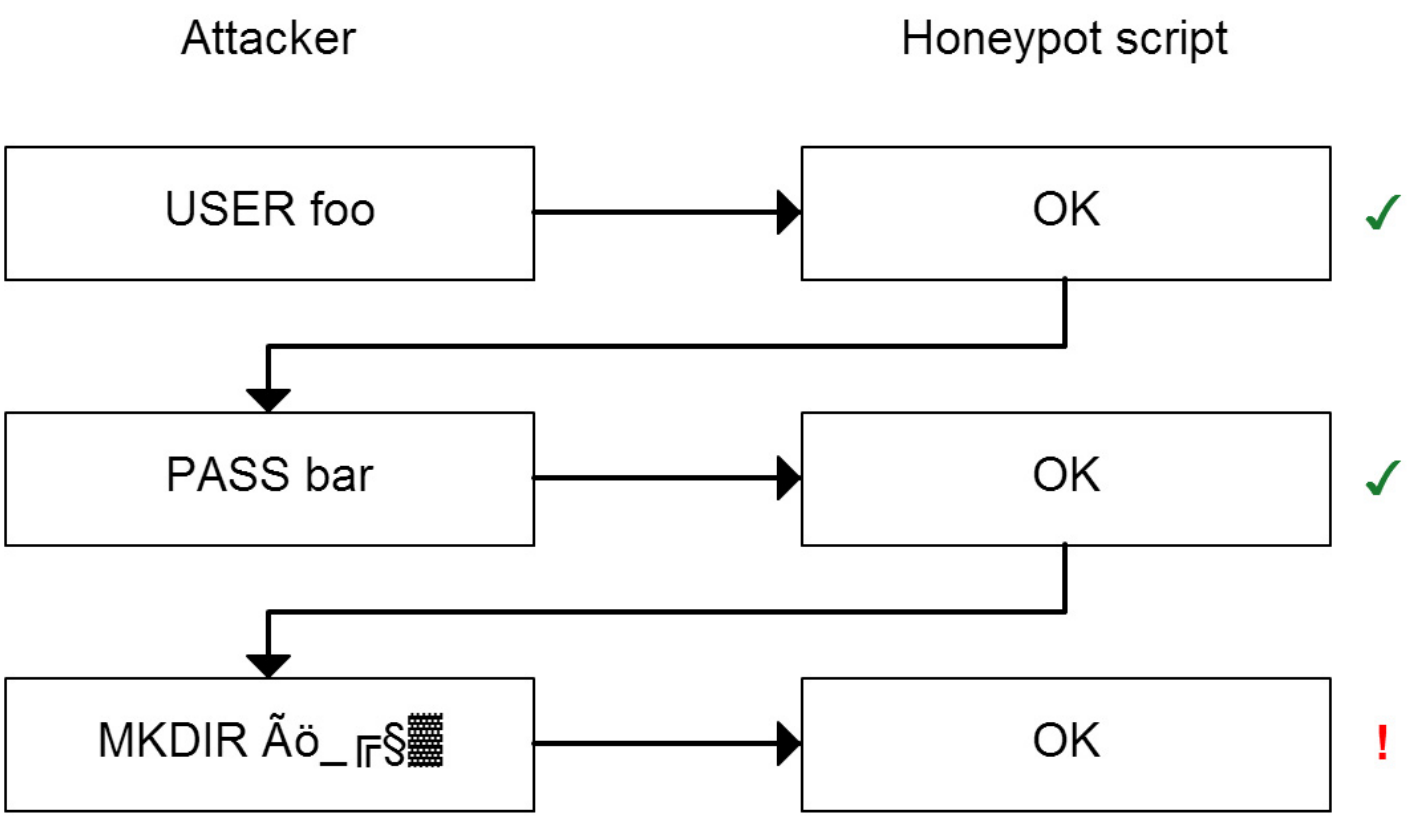
Process



Python honeypot script

- Small database with 30 vulnerable FTP banners for all 37 exploits
- Implemented responses to most used FTP commands
- Saves all traffic
- Detect "suspicious" traffic

Detect suspicious traffic



Extract signatures from suspicious traffic

- Collect multiple suspicious flows for the same exploit, different payload
- Find the longest string shared by all suspicious flows using the Longest Common Substring (LCS) algorithm
- The resulting string will be used as signature
- This method depends on static parts in the exploit, regardless of the payload

Extract signatures from suspicious traffic

Flow 1: **ffeeddcc**acbefafabcdefbafcbaedfeaf
Flow 2: aabcbeaf**ffeeddcc**afbdeaabcdefbcffea
Flow 3: feabcdefbfeacceafeabceffaecbeafabcaedd

The string "ffeeddcc" is the longest common substring in the first 2 flows, but it does not occur in the 3rd flow.

Extract signatures from suspicious traffic

Flow 1: ffeeddccacbefaf**abcdef**bafcbaedfeaf
Flow 2: aabcbeafffeeddccafbdea**abcdef**fcffea
Flow 3: fe**abcdef**afeacceafeabceffaecbeafabcaedd

The string "abcdef" is the longest common substring occurring in all flows. This will be the signature.

Extract signatures from suspicious traffic

LCS found "good" signatures for 20 exploits from their suspicious traffic flows. The rest either had no signature, or a too generic signature (e.g. "USER").

Solution: for the remaining exploits, run LCS on all other flows. Resulted in 12 "good" signatures for the remaining 17 exploits.

Matching signatures against traffic

With the signatures, we should be able to detect exploits:

- Check each incoming flow in the honeypot for known signatures
- If a signature is found, print out the matching exploit

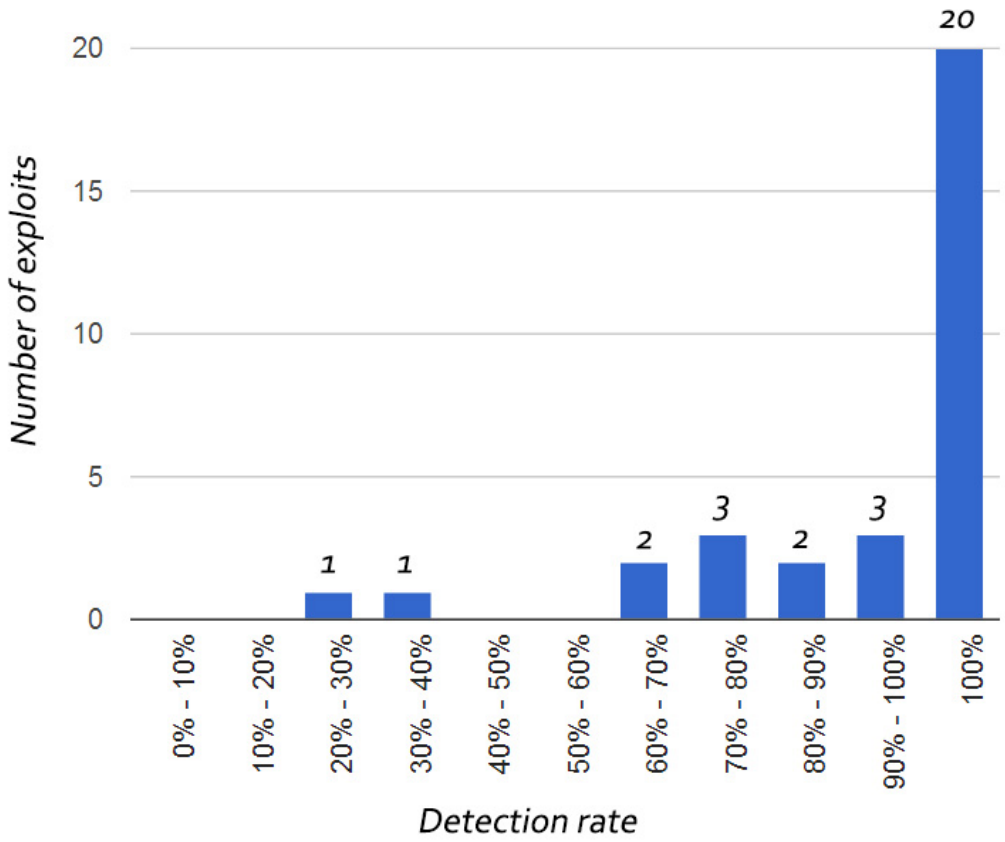
Matching signatures against traffic

Problem: some exploits share the same signature, causing false positives.
Easy solution: only check for signatures of exploits belonging to the current FTP banner.

Results

In total found signatures for 32 out of 37 exploits (86%). Test how good these signatures detect exploits by firing all exploits against the FTP honeypot script, with every possible payload.

Results



Average detection rate of 89.95%

Answering the research questions

How feasible is an automated method to detect specific exploits on a honeypot by monitoring network traffic of exploits?

- What setup is needed in order to have exploits successfully complete their exploit against a honeypot?
- What is the best method to process network traffic to/from the honeypot to extract and match a unique signature from exploit traffic?
- How successful are these methods?

Answering the research questions

What setup is needed in order to have exploits successfully complete their exploit against a honeypot?

Many of the exploits check FTP banner and correct FTP responses. In order to allow exploits to complete successfully, we need to emulate both the banner and the correct responses.

Answering the research questions

What is the best method to process network traffic to/from the honeypot to extract and match a unique signature from exploit traffic?

In this research, a granular method of storing and processing network traffic was used. Extract signatures using the LCS algorithm, match traffic against signatures on-the-fly proved very effective.

Answering the research questions

How successful are these methods?

Not all exploits yielded a signature, but for the exploits that did, most signatures have a high detection rate.

Answering the research questions

How feasible is an automated method to detect specific exploits on a honeypot by monitoring network traffic of exploits?

The methods presented work very well. Easily portable to other protocols/exploits. Can work standalone or as part of existing honeypot software.

Questions

Questions?

References

Anley, Chris, Heasman, John, Lindner, Felix, & Richarte, Gerardo. 2011.
The shellcoder's handbook: discovering and exploiting security holes.
Wiley.

Baumann, Reto, & Plattner, Christian. 2002.
White Paper: Honeypots.