Generating Fingerprints of Network Servers and their Use in Honeypots

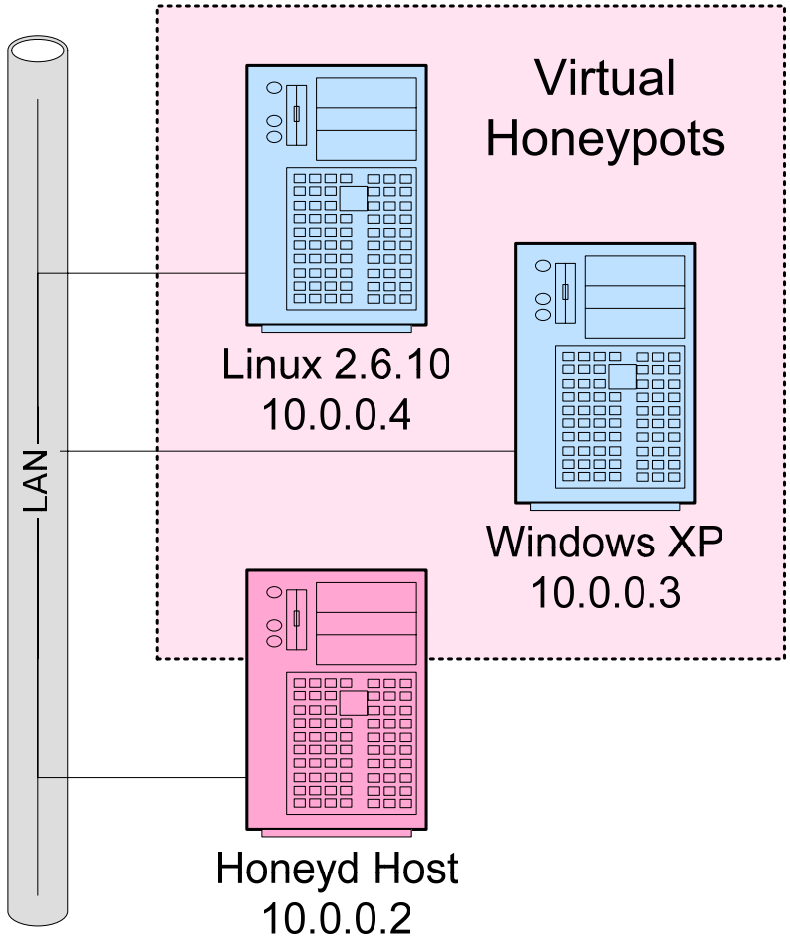
Thomas Apel

Introduction

- Fingerprinting of network servers
  - Banner string
  - Available options
  - Reactions to illegal syntax
- Usage in honeypots
  - Emulation of network servers
  - Is it possible to create such emulators automatically?

Honeyd

- Emulates hosts and network infrastructures
  - Emulates idiosyncrasies of different IP stacks
  - Deceives OS fingerprinting tools
  - Uses databases of Nmap, p0f, Xprobe2
- Add network services via:
  - External programs
  - Python plug-ins
  - Forwarding to real servers



Service Fingerprinting

- Identification possibilities
  - Banner strings
  - Advertisement of supported options
- FTP
  - TYPE command before login
  - CWD command without directory parameter
  - EPRT, EPSV supported?
- SMTP
  - Recipient address enclosed by angle brackets?

```
220 hostname.domain FTP server (Version
6.4/OpenBSD/Linux-ftpd-0.17) ready.

TYPE A
530 Please login with USER and PASS.

TYPE X
530 Please login with USER and PASS.
500 'TYPE X': command not understood.

TYPE C
530 Please login with USER and PASS.
500 'TYPE C': command not understood.
500 'TYPE C': command not understood.
```

Fingerprinting Tools

- Nmap
  - Covers lots of protocols
  - Scan limited to banners in most cases
  - Trigger commands if necessary
- Vmap
  - Covers 5 protocols
  - Database contains unchanged server responses
- Amap
  - Covers many protocols
  - Sends simple trigger commands
- Smtpscan
  - Only 15 tests
  - Evaluates only status codes
  - More than 3000 fingerprints
- Ftpmap
  - Extensive tests
  - How random are port assignments?
  - About 70 fingerprints
- And others…

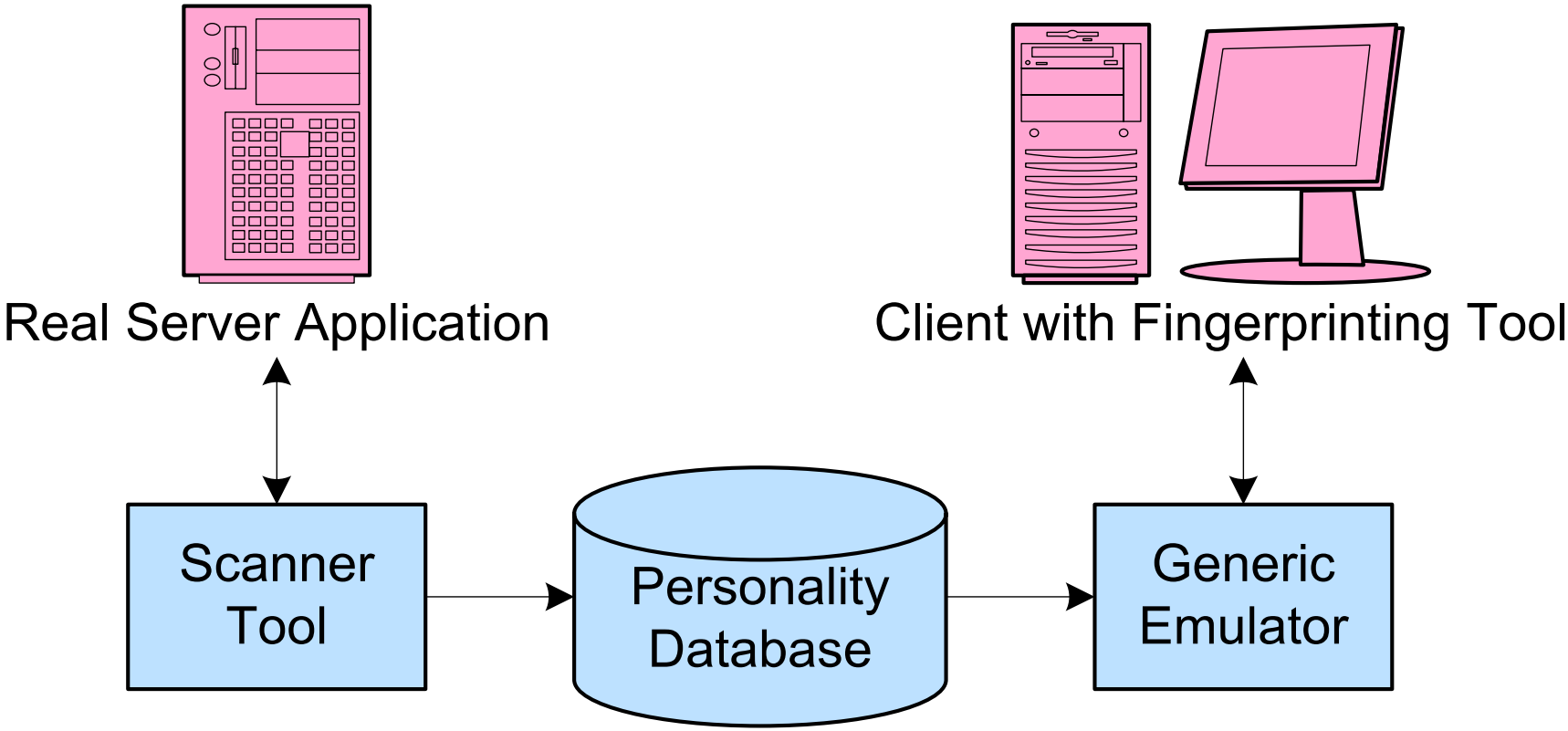
Existing Service Emulators

- IIS Emulator
  - Emulates Microsoft's Internet Information Server
  - Noticeable feature set
- smtp.pl
  - Comes with Honeyd
  - Offers Sendmail and Postfix personalities
  - Should be possible to add additional personalities easily
- Other emulators
  - Depth of emulation very limited
  - Do not always fool fingerprinting tools
  - Personalities not interchangeable
  - All existing emulators seem to be handcrafted solutions

Problem and Idea

- Problem
  - Most existing emulators are rather simple
  - Their manual creation is time consuming
- Idea
  - Automatically create server emulators
  - At least give assistance in the creation process

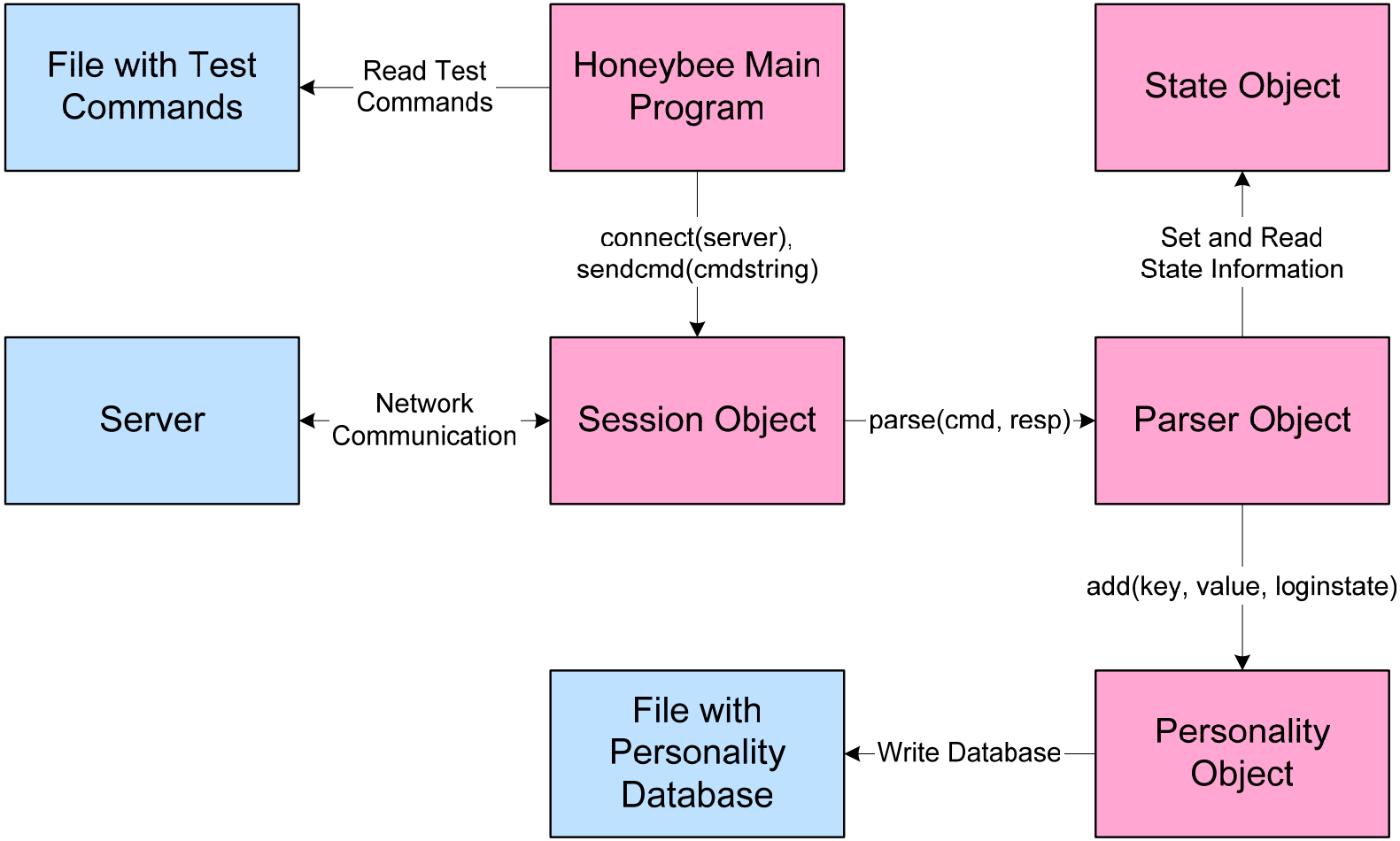
Concept


Real Server Application
Client with Fingerprinting Tool
Scanner Tool
Personality Database
Generic Emulator

Reducing Complexity

- Problem
  - Number of possible command strings is infinite
  - Possibly infinite number of server states
  - How to control generic emulator?
- Solution
  - Focus on most important server states
  - Put command strings into classes and test only representatives
  - Error codes of response give sufficient control information
- Example SMTP
  - Do not test arbitrary recipient addresses
  - Test reaction with known user, unknown user, and remote user

Honeybee Scanner



Parsing Functions

* D:\Workspace\HoneyBee\honeybee\smtp.py - Notepad2

Datei Bearbeiten Ansicht Einstellungen ?

```
182     def parse_rcpt(self, cmd, args, resp):
183         """Parse responses for RCPT command."""
184         [...]
185         elif args.upper().startswith("TO:"):
186             rcpt = args
187             rcpt = rcpt[3:] # strip "TO:"
188             rcpt = rcpt.strip()
189
190             bracketstyle = self.bracketstyle(rcpt)
191             rcpt = self.removebrackets(rcpt)
192
193             if rcpt == self.state.config["knownuser"]:
194                 rcptclass = "<knownuser>"
195             elif rcpt == self.state.config["unknownuser"]:
196                 rcptclass = "<unknownuser>"
197             elif rcpt == self.state.config["relayaddress"]:
198                 rcptclass = "<relayaddress>"
199             [...]
200             if resp[:3] == "250":
201                 self.state.seenrcpt = True
202             resp = resp.replace(rcpt, "%(rcpt)s")
203             key = cmd + " TO: " + rcptclass + " " + bracketstyle
204         [...]
205         return key, resp
```

Personality Database

- Hash table
  - Stored in text file
  - Lines with tuples of key and value
  - Conversion via Python’s repr() and eval() functions
- Keys
  - Commands
  - Parameters
  - State information
- Values
  - Server response
  - Placeholders for dynamic parts

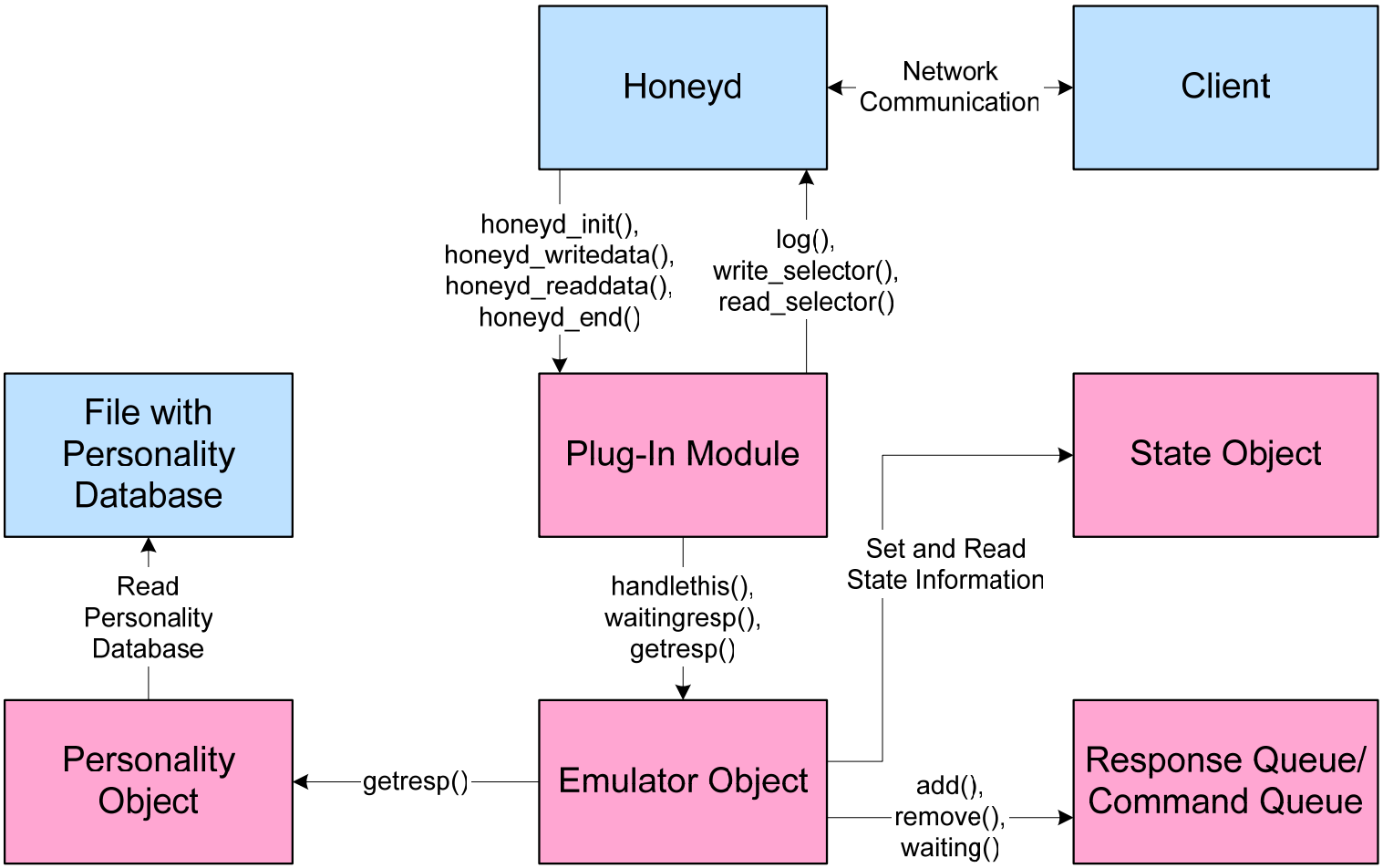
Personality Database – Example

* D:\Workspace\HoneyBee\honeyd-plugin\def\smtp\sendmail-8.13.1-freebsd - Notepad2

Datei Bearbeiten Ansicht Einstellungen ?

```
1  ('RCPT TO: <relayaddress> withoutbrackets', '550 5.7.1 %(rcpt)s... Relaying denied
2  ('RCPT TO: <relayaddress> startbracketmissing', "553 5.0.0 %(rcpt)s>... Unbalanced
3  ('RCPT TO: <relayaddress> endbracketmissing', "553 5.0.0 <%(rcpt)s... Unbalanced '
4  ('RCPT TO: <relayaddress> withbrackets', '550 5.7.1 <%(rcpt)s>... Relaying denied\
5
6  ('RCPT TO: <knownuser> withoutbrackets', '250 2.1.5 %(rcpt)s... Recipient ok\r\n')
7  ('RCPT TO: <knownuser> startbracketmissing', "553 5.0.0 %(rcpt)s>... Unbalanced '>
8  ('RCPT TO: <knownuser> endbracketmissing', "553 5.0.0 <%(rcpt)s... Unbalanced '<'\
9  ('RCPT TO: <knownuser> withbrackets', '250 2.1.5 <%(rcpt)s>... Recipient ok\r\n')
10
11 ('RCPT TO: <unknownuser> withoutbrackets', '550 5.1.1 %(rcpt)s... User unknown\r\n
12 ('RCPT TO: <unknownuser> startbracketmissing', "553 5.0.0 %(rcpt)s>... Unbalanced
13 ('RCPT TO: <unknownuser> endbracketmissing', "553 5.0.0 <%(rcpt)s... Unbalanced '<
14 ('RCPT TO: <unknownuser> withbrackets', '550 5.1.1 <%(rcpt)s>... User unknown\r\n'
15
16 ('RCPT', '501 5.5.2 Syntax error in parameters scanning ""\r\n')
17 ('RCPT nosender', '503 5.0.0 Need MAIL before RCPT\r\n')
```

Emulator Plug-In for Honeyd



Command Handlers

* D:\Workspace\HoneyBee\honeyd-plugin\smtp.py - Notepad2

Datei Bearbeiten Ansicht Einstellungen ?

```
218     def handle_rcpt(self, cmd, args):
219         """RCPT command handler."""
220         if self.state.hassender():
221             if args.upper().startswith("TO:"):
222                 [...]
223                 bracketstyle = self.bracketstyle(rcpt)
224                 rcpt = self.removebrackets(rcpt)
225                 # Check if recipient is in user list.
226                 [...]
227                 if rcpt in userlist:
228                     rcptclass = "<knownuser>"
229                 else:
230                     rcptclass = "<unknownuser>"
231                 if rcpt != "":
232                     key = cmd + " TO: " + rcptclass + " " + bracketstyle
233                 else:
234                     key = cmd + " TO:"
235                 resp = self.pers.getresp(key)
236                 # Store recipient only on error code 250.
237                 if resp[:3] == "250":
238                     self.state.addrcpt(rcpt)
239                 resp = resp % {"rcpt": rcpt}
240         [...]
241         return resp
```

Pros and Cons

- Pros
  - Approach is straightforward
  - Saves costs when creating emulators of many different servers
  - Sufficient for deceiving current fingerprinting tools
- Cons
  - Scanner must fit server application
  - Creation of test sets and parser function is still daunting task
- Missing functionality in Honeyd
  - Service emulators cannot initiate/accept connections
  - Honeyd cannot pass configuration data to Python plug-ins

Emulations vs. Fingerprinting Tools

- Emulated servers
  - Sendmail 8.13.1
  - Postfix 2.1.5
  - Exim 4.44
  - FreeBSD 5.3 FTPd
  - OpenBSD FTPd 6.4 (Linux)
  - Pure-FTPd 1.0.19
  - Qpopper 4.0.5
- Fingerprinting Tools
  - Nmap 3.75
  - Amap 4.8
  - Vmap 0.6
  - Smtpscan 0.5
  - Ftpmap 0.5

Test Results (1)

- Nmap
  - Perfect results
- Amap
  - Falsely identifies FTP servers as SMTP servers
    - Applies to both emulations and original servers
  - Problem with Sendmail emulation
    - Properly recognized as mail server
    - But additionally mistaken for X Window System, Oracle DB, and others
    - Cause is handling of binary commands
    - Python strips unprintable chars, with C null terminates string

Test Results (2)

- Vmap
  - Match rates are only difference in most cases
  - They are slightly lower for emulations
  - Problem with Postfix
    - Emulation is properly recognized
    - But original server aborts session because of too many errors
- Smtpscan
  - Perfect results
- Ftpmap
  - Error rates are only difference between emulation and original
  - They are slightly higher for emulations

Summary and Conclusion

- Main problem:
  - Complexity
- It works:
  - Current version covers SMTP, FTP, and POP3
  - Emulations deceive current fingerprinting tools
- But:
  - Creation process not always fully automatic
  - New target servers might require changes to parsers and command handlers
  - Providing real functionality for complex protocols like FTP still difficult
- Benefit depends on:
  - How many different emulators are needed?
  - How similar are the emulation targets?