Generating Fingerprints of Network Servers and their Use in Honeypots

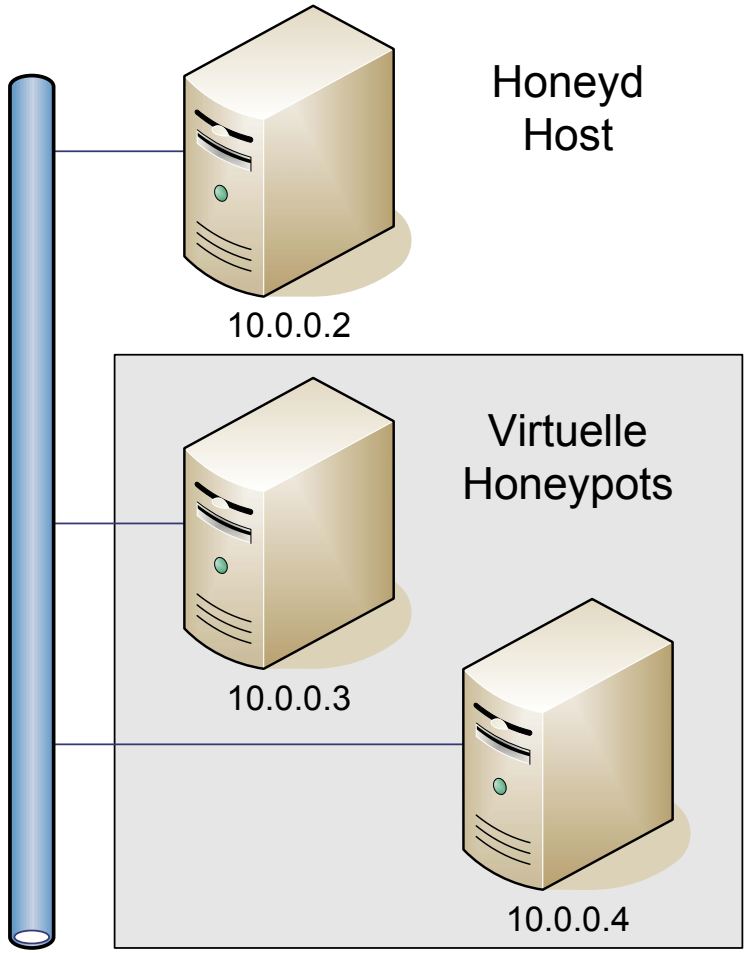
Thomas Apel

Der Überblick

- Fingerprinting von Netzwerkdiensten
  - Banner
  - Verfügbare Optionen
  - Reaktionen auf falsche Syntax
- Verwendung für Honeypots
  - Emulator für Netzwerkdienste
  - Soll automatisch erzeugt werden
  - Plug-In für Honeyd

Honeyd

- Emuliert Rechner und Infrastruktur
  - Emuliert Eigenheiten der IP-Stacks
  - Täuscht damit Netzwerkscanner
  - Benutzt Datenbanken von Nmap, p0f, Xprobe2
- Netzwerkdienste
  - Shellskripte
  - Pythonmodule
  - Weiterleitung an echte Server



Fingerprinting-Tools

- Nmap
  - Viele Protokolle
  - Prüft Banner
- Vmap
  - Viele Protokolle
  - Datenbank enthält unveränderte Antworten
- Amap
  - Viele Protokolle
  - Trigger für z.B. DNS
- Smtpscan
  - Nur 15 Tests reichen
  - Nur Statuscodes werden ausgewertet
  - Über 3000 Fingerprints
- Ftpmap
  - Ausführliche Tests
  - Testet Zufälligkeit der Portauswahl
  - Knapp 70 Fingerprints
- Und noch ein paar andere…

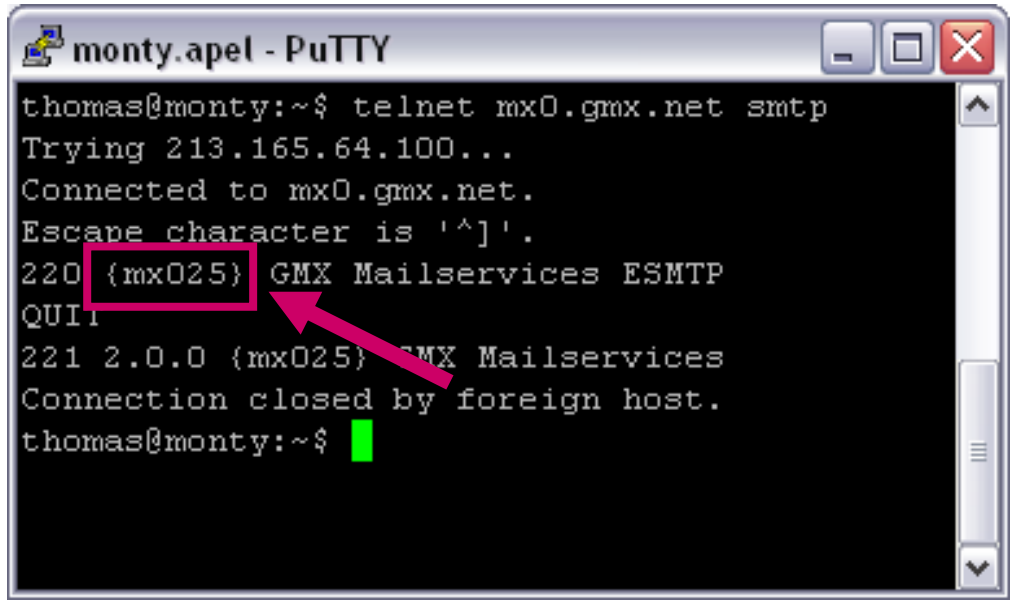
Beispiele: FTP

- TYPE Kommando ohne Login
  - FreeBSD FTPd
  - Linux Netkit FTPd
- CWD ohne Parameter
  - In RFC nicht definiert
  - Wechsel zu Root-Directory
  - Bleibe im aktuellen Verzeichnis
- Werden EPRT und EPSV unterstützt?
- Negative Integer als Parameter für PORT, ALLO, etc.

```
220 freebsd.domain FTP server (Version
6.00LS) 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.


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

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.
```

Beispiele: SMTP

- Tests aus Smtpmap
  - Sind Exoten wie SOML, SAML oder TURN implementiert?
  - Mail-Adresse mit nur einer spitzen Klammer
- Server von GMX
  - Geschweifte Klammern
  - Kein FQDN
  - Damit überhaupt RFC-konform?



Existierende Emulatoren

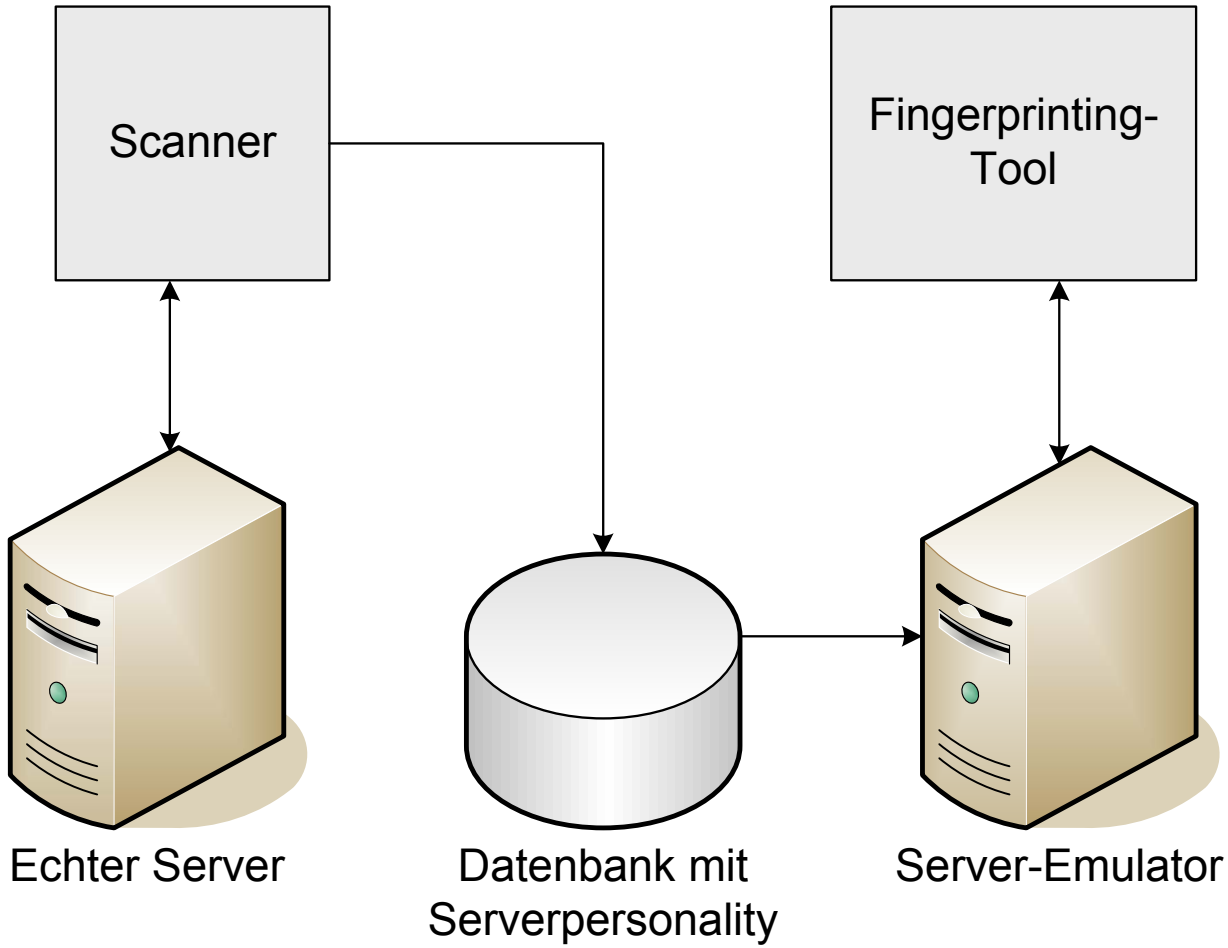
- IIS Emulator
  - Emuliert Microsofts Internet Information Server
- smtpot.py
  - Emuliert Sendmail
- smtp.pl
  - Liegt Honeyd bei
  - Emuliert Sendmail und Postfix
  - Kann begrenzt um andere Server erweitert werden
- Probleme
  - Emulieren nur bestimmte Serverinkarnationen
  - In Handarbeit entstanden

Der Wunsch

Der Plan

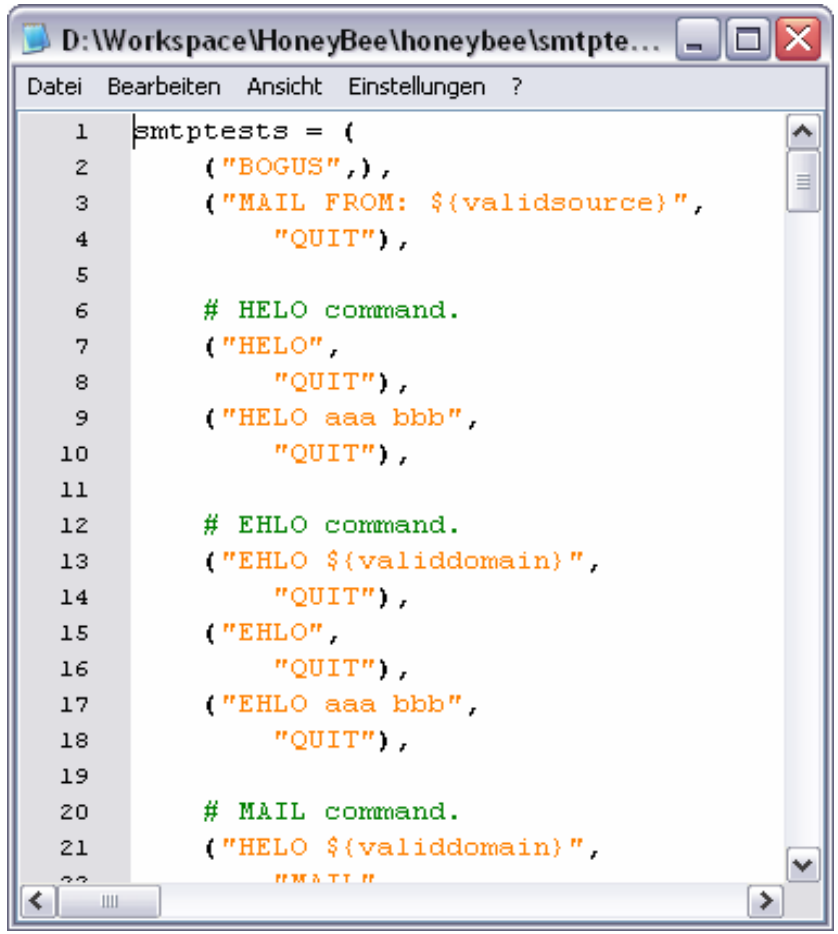
- Scanner
  - Unterhält sich mit Server
  - Testet Reaktion in verschiedenen Situationen
  - Generiert Datenbank
- Datenbank
  - Enthält Antworten für verschiedene Situationen
- Emulator
  - Findet Antworten in Datenbank
  - Geschrieben in Python
  - Nutzt Plugin-Schnittstelle von Honeyd

Das große Ganze

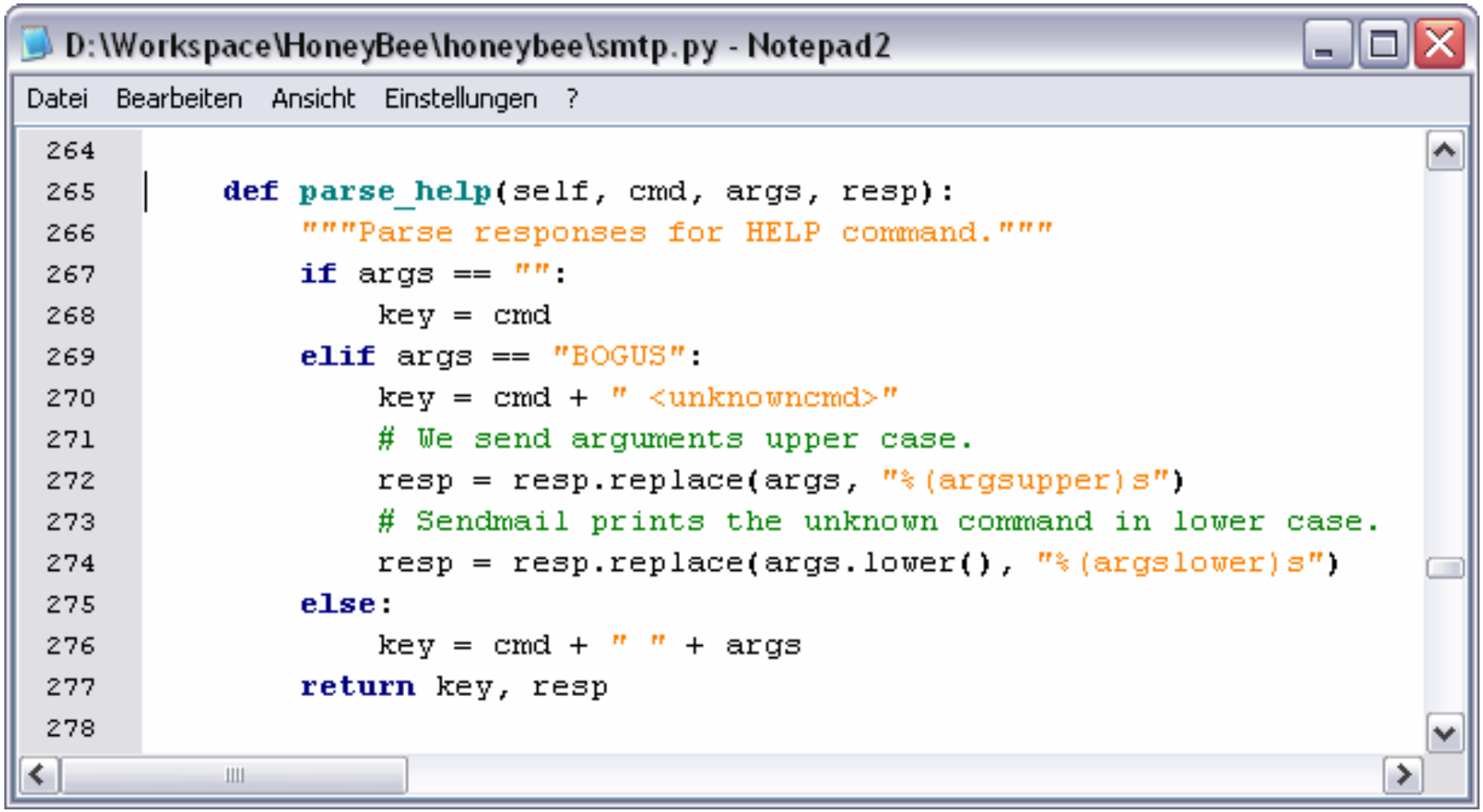


Der Scanner

- Provoziert verschiedene Situationen und speichert Antworten
- Parser pro Kommando
  - Dynamische Teile werden durch Platzhalter ersetzt
  - Falsche Parameter werden zu Klassen zusammengefasst
- Scanner muss sich seinen Zustand merken
  - Login
  - Wurde Absender gesetzt?
  - Wurde Empfänger gesetzt?



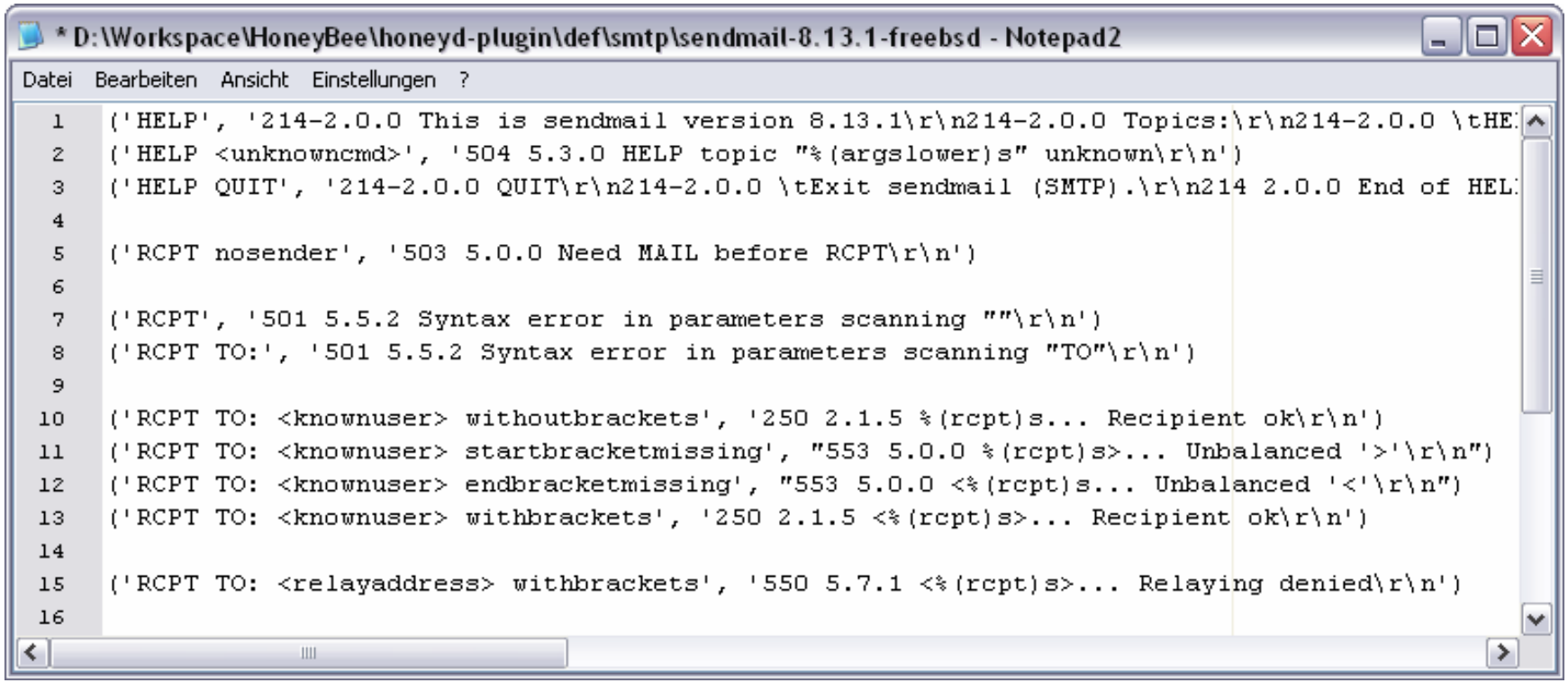
Der Scanner


D:\Workspace\HoneyBee\honeybee\smtp.py - Notepad2
Datei Bearbeiten Ansicht Einstellungen ?
264
265 def parse_help(self, cmd, args, resp):
266 """Parse responses for HELP command."""
267 if args == "":
268 key = cmd
269 elif args == "BOGUS":
270 key = cmd + " <unknowncmd>"
271 # We send arguments upper case.
272 resp = resp.replace(args, "%(argsupper)s")
273 # Sendmail prints the unknown command in lower case.
274 resp = resp.replace(args.lower(), "%(argslower)s")
275 else:
276 key = cmd + " " + args
277 return key, resp
278

Die Datenbank

- Hashtabelle/Dictionary
  - Gespeichert in Textfile
  - Zeilenweise Tupel aus Schlüssel und Wert
  - Einlesen und speichern per repr() und eval()
- Schlüssel
  - Kommandos
  - Parameter
  - Situation bzw. Zustand
- Wert
  - Serverantwort
  - Mit Platzhaltern für dynamische Antwortteile

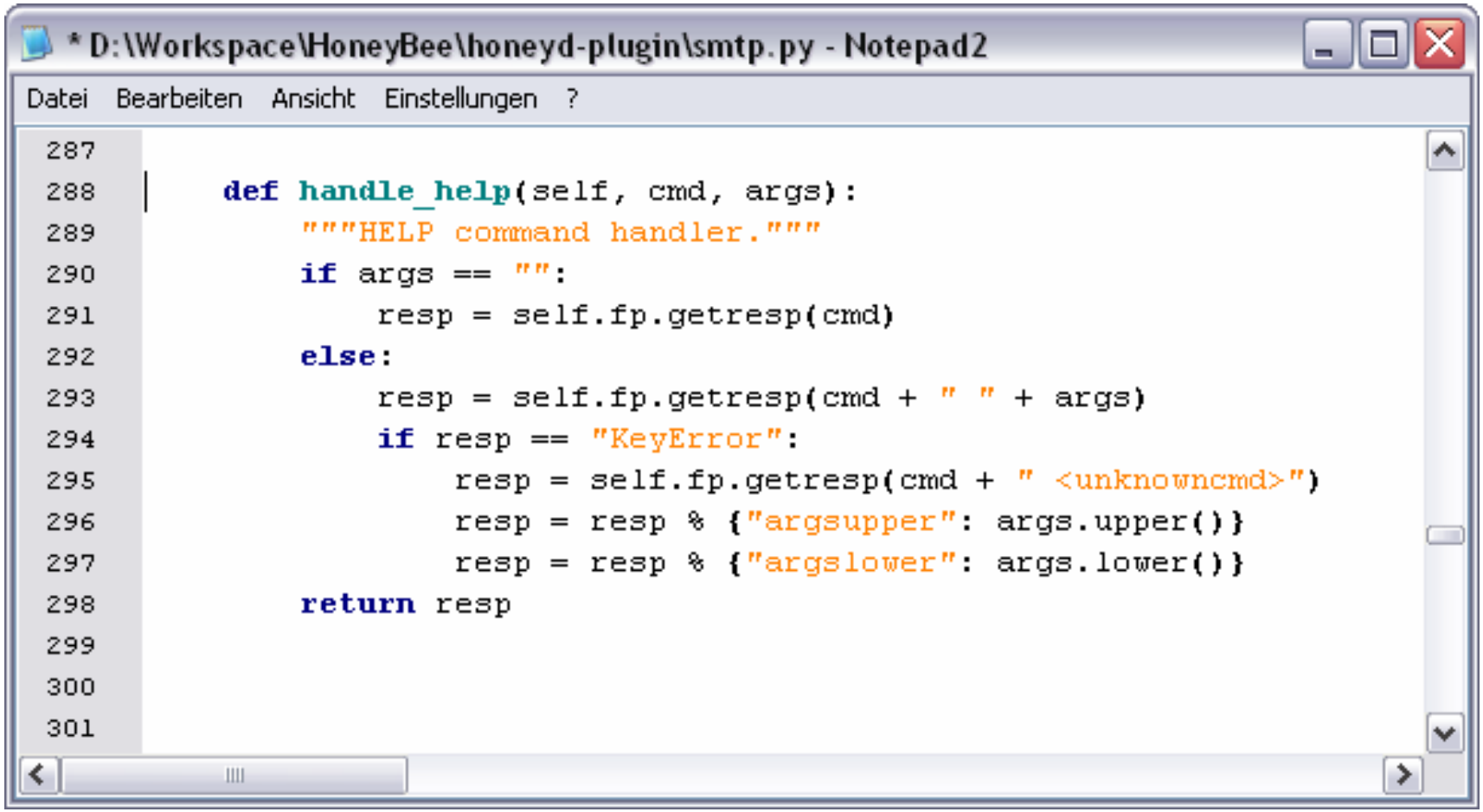
Die Datenbank



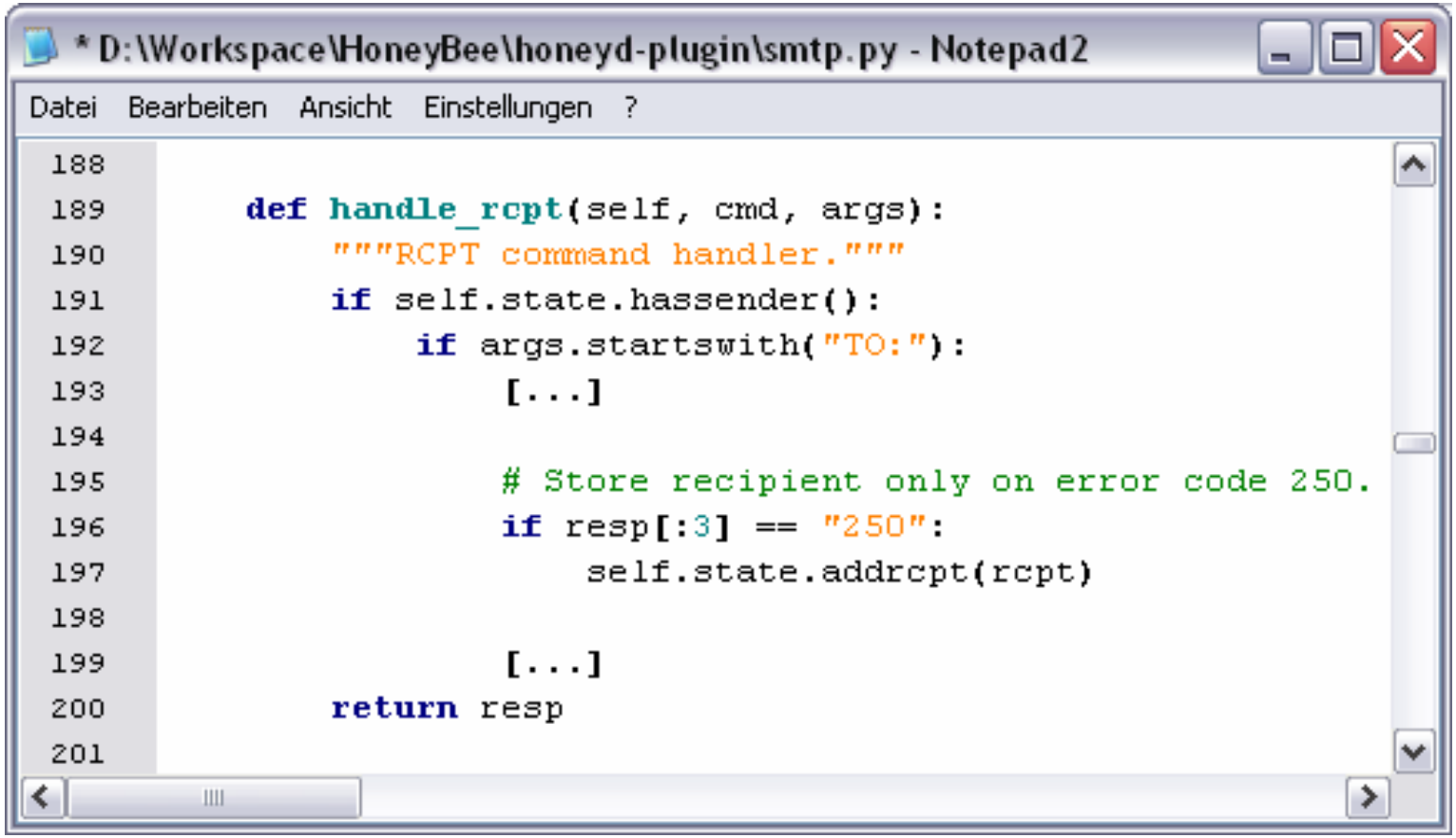
Der Emulator

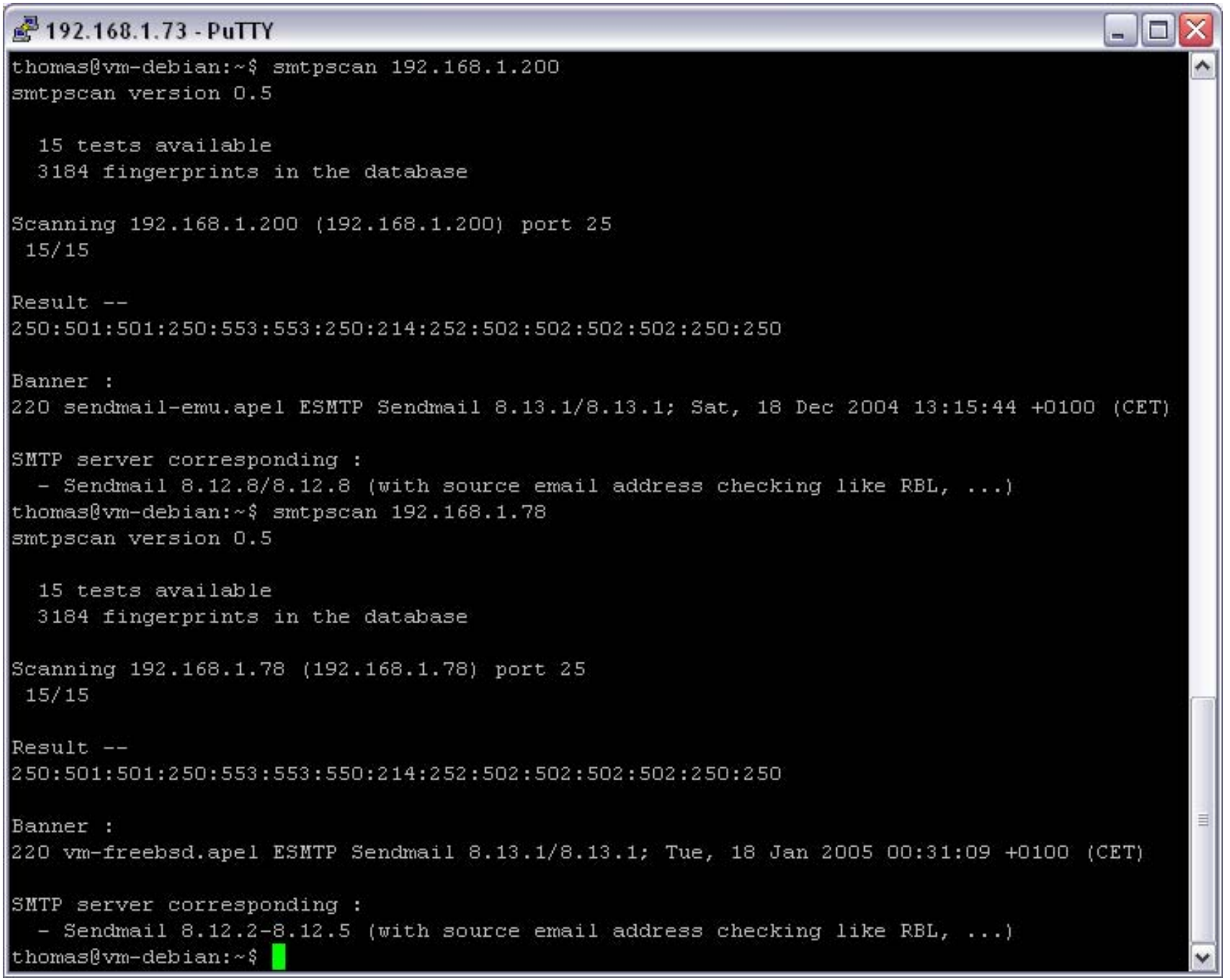
- Anfragen werden in Kommando und Parameter zerlegt
- Command-Handler
  - Analysiert Anfrage
  - Berücksichtigt Zustand
  - Holt Antwort aus Datenbank
  - Ersetzt Platzhalter mit dynamischen Daten
- Steuerung
  - Übergang von einem Zustand zum nächsten
  - Zum Beispiel: Wurde Absender mit fehlender spitzer Klammer akzeptiert?
  - Fehlercodes der Antwort reichen

Der Emulator



Der Emulator




192.168.1.73 - PuTTY
thomas@vm-debian:~$ smtpscan 192.168.1.200
smtpscan version 0.5
15 tests available
3184 fingerprints in the database
Scanning 192.168.1.200 (192.168.1.200) port 25
15/15
Result --
250:501:501:250:553:553:250:214:252:502:502:502:502:250:250
Banner :
220 sendmail-emu.apel ESMTP Sendmail 8.13.1/8.13.1; Sat, 18 Dec 2004 13:15:44 +0100 (CET)
SMTP server corresponding :
- Sendmail 8.12.8/8.12.8 (with source email address checking like RBL, ...)
thomas@vm-debian:~$ smtpscan 192.168.1.78
smtpscan version 0.5
15 tests available
3184 fingerprints in the database
Scanning 192.168.1.78 (192.168.1.78) port 25
15/15
Result --
250:501:501:250:553:553:550:214:252:502:502:502:502:250:250
Banner :
220 vm-freebsd.apel ESMTP Sendmail 8.13.1/8.13.1; Tue, 18 Jan 2005 00:31:09 +0100 (CET)
SMTP server corresponding :
- Sendmail 8.12.2-8.12.5 (with source email address checking like RBL, ...)
thomas@vm-debian:~$

Die Schwierigkeiten

- Scanner muss zu Servern passen
  - Dynamische Stellen müssen erkannt und ersetzt werden
  - Zusätzliche Tests erfordern Anpassung der Parser
- Komplexität
  - Immer noch jede Menge Handarbeit
  - Echte Funktionalität, z.B. bei FTP
- Fehlende Funktionen von Honeyd
  - FTP-Datenverbindungen nicht möglich
  - Keine zentrale Konfiguration möglich

Das Fazit

- Das Prinzip funktioniert
  - SMTP täuscht Fingerprinter und nimmt Mails an
  - FTP täuscht Fingerprinter
- Mögliche Erweiterungen und Verbesserungen der Implementierung
  - POP3 ist noch in Arbeit
  - Sonstige Protokolle wie IMAP4, SSH?
  - Zusätzliche Tests
  - Mehr Funktionalität für Emulation