HoneySpam: Honeypots fighting SPAM at the source

Mauro Andreolini
University of Modena
andreolini@unimore.it

Alessandro Bulgarelli
University of Modena
bulgarelli.alessandro@unimore.it

Michele Colajanni
University of Modena
colajanni@unimore.it

Francesca Mazzoni
University of Modena
mazzoni.francesca@unimore.it
Outline

- Receiver-oriented anti-SPAM tools
- Source-oriented anti-SPAM tools
- Requirements of an anti-spam system
- HoneySpam architecture
- HoneySpam emulated services
- Conclusions and future work
The growth of SPAM traffic

Source: http://www.stilgherrian.com/spam/
Receiver-oriented anti-SPAM tools

- Most anti-SPAM tools are receiver-oriented
- Proper filtering actions are taken after the delivery of the message
  - at the server level
    - Sophos MailMessage, MailSWAT, MailStripper
  - at the client level
    - Sophos MailMonitor, WebWasher
  - at both levels
    - SpamAssassin
- Still provide false negatives
- Do not aim at reducing unwanted Internet traffic
Source-oriented anti-SPAM tools

- Try to fight SPAM acting on the SPAM sources
  - Examples: SMTP server black/white lists

- Issues with black lists:
  - brute force approach, does not scale with the increasing number of spammers
  - black lists do not help in reducing unwanted traffic

- Issues with white lists:
  - really effective for specific user communities
Spammer activities

- Sending unsolicited e-mails is just the last step of a complex series of operations:
  - crawling Web sites for e-mail harvesting
  - search and use of open proxies to operate anonymously
  - search and use of open relays to send e-mails without need for authentication

Remarks
- Different actions call for different tools
- Fight these actions at their source
- Try to reduce unwanted network traffic
Our goal

- Present a framework of tools that:
  - provides attracting services to spammers
  - fights spamming activities at their sources
  - tries to reduce unwanted network traffic related to unsolicited e-mail messages
  - is fully compliant with existing protocols and practices
Requirements of an anti-SPAM system

- **Reduce the efficiency of crawlers**
  - force crawlers into an endless loop
  - e-mail address database poisoning
  - protect legitimate crawlers

- **Identify spammers**
  - log every spammer activity

- **Block spam e-mails**
  - must not block valid e-mail messages (false positives)
  - should pass the least amount of unsolicited messages (false negatives)
HoneySpam: implementation details

- The emulated services are implemented through the *honeyd* daemon
  - emulates operating system TCP/IP stacks
  - emulates common servers (Web, SMTP) through Perl scripts
  - easy to setup (through one relatively simple configuration file)
  - low overhead

- Configuration personality
- Packet dispatcher
- Personality engine
HoneySpam: services

- GOAL: hinder the work of illegitimate crawlers
- E-mail database poisoning
  - automatic building of HTML pages with fake e-mail addresses
- Crawler slowdown
  - automatic generation of endless link loops that block crawlers
- Compliance with legitimate crawlers
  - implements the robot exclusion protocol
- Spammers traceback
- Logging of client requests

Steps to Reducing Unwanted Traffic on the Internet (SRUTI 2005)
HoneySpam: services

Emulated Open Proxy

**GOAL:** identify spammers trying to operate through open proxy chains

- emulate a subset of the HTTP protocol
- redirection of HTTP proxy CONNECT requests to port 25 towards an emulated open relay
- HTTP proxy CONNECTs to other ports are answered with an error message
- logging of client requests
HoneySpam: services

Emulated Open Relay

● **GOAL:** block the traffic associated to unsolicited e-mail messages
● emulates postfix/sendmail MTA
● e-mails are not delivered, but saved for later analysis
  • actually, the first e-mail is also sent to let the spammer believe that the service is working
● logging of client activity
HoneySpam: implementation details

● Emulated OSs:
  - FreeBsd, Linux (2.4, 2.6 kernel), Windows 2000 and others (through nmap, xprobe2 and p0f fingerprints)

● Emulated services:
  - Web servers: Apache, IIS
  - SMTP servers: Postfix, Sendmail
  - Proxy servers: SOCKS4/5-based servers

● Emulated routers:
  - Cisco, Zyxel, Intel, 3Com
Possible attacks to HoneySpam

• Honeypot identification
  • Not vulnerable to:
    • network scanners (nmap, xprobe2, p0f)
  • Vulnerable to:
    • service scanners (honeypot hunter)
    • black list services

• Intrusion
  • Not vulnerable to:
    • remote attacks (if chrooted/jailed)
  • Vulnerable to:
    • honeyd exploits
Conclusions

- Implementation of a framework for fighting SPAM at the source
  - Reduce the associated traffic
  - Reduce the effectiveness of spamming techniques

- Emulated services:
  - Web server
    - pollution of spammer databases
    - slowdown and blocking of illegitimate crawlers
  - Open Proxy
    - spammers trace-back
    - redirection of spammer requests to emulated open relays
  - Open Relay
    - block the traffic associated to unsolicited messages

- Logging of spammer activity
Future work

- Scalability
  - Geographical replication of the framework
  - Clustering of HoneySpam in a LAN

- Fault-tolerance
  - If HoneySpam is detected, it is no longer useful
  - Many running HoneySpam instances make detection and black-listing harder

- Limiting the network throughput of spammers
  - Bandwidth-limiting traffic related to spamming activities
Future work

- Collaborative environment: extend HoneySpam to allow information exchange
- Sources of information exchange:
  - remote HoneySpams
  - authorized SMTP servers
  - Open proxy lists
  - Web server log information pertaining illegitimate crawlers (name, IP address)
Thanks for your attention