

Dynamic Application-Layer Protocol Analysis

For Network Intrusion Detection

Holger Dreger, TU München

Anja Feldmann, T-Labs / TU Berlin

Michael Mai, TU München

Vern Paxson, ICSI / LBNL

Robin Sommer, ICSI

NIDS: State-of-the-Art

- Protocol-specific traffic analysis
 - ⇒ Semantic context for (much) better detection quality
- How to decide which protocol to analyze?
 - Relies on well-known port numbers
(e.g., HTTP if-and-only-if TCP port 80)
(or um maybe 8080 and 8000 and)
- And if it's *not* on a well-known port?
 - Perhaps use byte-level signatures to *flag* what protocol it appears to be

Problem

- Applications use arbitrary ports!
 - Benign reasons
 - Lack of user privileges, obfuscation, multiple versions
 - Adversarial applications (maybe not so benign), e.g., Skype bypassing firewalls
 - Malicious intent
 - Evasion of security monitoring
 - E.g., IRC-botnets on ports other than 666x/tcp
 - E.g., Pirate FTP-servers on ports other than 21/tcp
- How to distinguish these?

Structure

- Prevalence of the problem
- Approach for dynamic analysis in NIDS
- Applications of new capabilities
- Performance evaluation

Prevalence of the Problem

- Data:
 - 24 hour full packet trace from MWN
 - 3.2 TB of data in 6.3 billion pkts,
137M TCP connections
 - Successful TCP connections: ~78%
 - Successful TCP connections on unpriv. Port: ~4%

Protocol Detection - Alternatives

- Statistical approach
 - E.g., packet size distribution
 - Suitable for separating interactive/bulk traffic
 - E.g., [Zhang00], [Moore05]
- Detect protocol patterns
 - Signatures (already implemented e.g., Linux netfilter I7-filter)
 - Maybe: Protocol detection by plausibility heuristics

Prevalence of the Problem

- Data:
 - 24 hour full packet trace from MWN
 - 3.2 TB of data in 6.3 billion pkts, 137M TCP connections
 - Successful TCP connections: ~78%
 - Successful TCP connections on unpriv. Port: ~4%
- Application protocol signatures from Linux netfilter I7-filter system
- Focus on HTTP, FTP, IRC, SMTP

Protocol Detection: Signatures

Protocol	HTTP	IRC	FTP	SMTP
Method				
Port (succ.)	93 429K	75.9K	151.7K	1 447K
Signature	94 326K	74.0K	125.3K	1 416K

Protocol Detection: Signatures

Protocol	HTTP	IRC	FTP	SMTP
Method				
Port (succ.)	93 429K	75.9K	151.7K	1 447K
Signature	94 326K	74.0K	125.3K	1 416K
expected port	92 228K	71.5K	98.0K	1 415K
other port	2 126K	2.5K	27.3K	0.3K

- Most (*but not all*) successful connections trigger expected signature
- FTP: high percentage of false negatives
- „Other port“ matches: needs further investigation

Protocol Signatures: Well-known Ports

Protocol Port	HTTP	IRC	SMTP	Other	No match
80	92 228 291				
666x		71 650			
25			1 415 428		

Protocol Signatures: Well-known Ports

Protocol Port	HTTP	IRC	SMTP	Other	No match
80	92 228 291	59	0	41 086	1 158 977
666x	1 217	71 650	0	4 238	524
25	459	2	1 415 428	195	31 889

- Some connections trigger more than one signature
- Some inappropriate use of well known ports

Observations

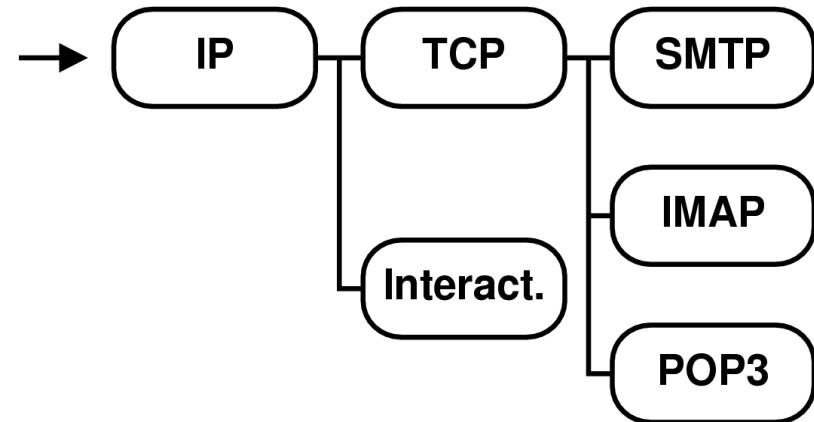
- Imprecision of signatures:
 - False negatives highlight need for refined signatures and/or *more context*
 - False positives (e.g., multiple matches for single connection) highlight limits in discriminating power
- Problem is real:
 - If we just believe port numbers, *numerous misidentifications*

Structure

- Prevalence of the problem
- **Approach for dynamic analysis in NIDS**
- Applications of new capabilities
- Performance evaluation

Approach for Dynamic Analysis

- Goals
 - Detection scheme independence
 - Dynamic **analysis**
 - Greater precision
 - Modularity
 - Efficiency
 - Customizability
- Idea: match multiple possible protocols **in parallel**
- Design (*see paper*)
 - Dynamic processing path
 - Per connection dynamic analyzer tree
 - Buffer connection data



Implementation

- Implemented in Bro NIDS
 - New „Protocol Identification Analyzer“ (PIA) implements protocol-detection and buffering
 - Required changing Bro's notion of one-to-one static binding from transport analyzer to application analyzer(s)
- Running in three large environments:
 - MWN, UCB, and LBNL

Deployment Trade-Offs

- Protocol detection signatures
 - Loose signatures affordable (false positives **fixed later**)
 - (Plus: improve accuracy w/ *bidirectional* signatures)
- When to decide whether candidate protocol actually in use?
 - **Positive**: Successful parse + threshold for volume, time
 - **Negative**: Unsuccessful parse + user control over whether to actually give up

Structure

- Prevalence of the problem
- Approach for dynamic analysis in NIDS
- **Applications of new capabilities**
- Performance evaluation

Reliable Real-Time Protocol Detection on non-Standard Ports

	Internal	Remote
FTP servers	6	17
HTTP servers	568	54,830
IRC servers	2	33
SMTP servers	8	8

- 1 day at UC Berkeley (MWN similar)

Reliable Real-Time Protocol Detection on non-Standard Ports

	Internal	Remote
FTP servers	6	17
HTTP servers	568	54,830
IRC servers	2	33
SMTP servers	8	8

- 1 day at UC Berkeley (MWN similar)
- Connections on non-standard ports mainly HTTP
 - UCB: split between real HTTP (e.g., Apache) & Gnutella
 - MWN: Similar, but more P2P (BitTorrent), also some FTP
 - Open HTTP proxies detected and closed
 - Open SMTP relay detected and closed

Payload Inspection of FTP Data Transfers

- FTP data transfers use arbitrary ports
 - Identify based on prior PORT, PASV
- Check connection payload using *libmagic*
 - Actual file type == expected file type?
 - E.g., could find rootkit tarball sent in .jpg
- Extension:
 - Use same mechanism for SMTP (mail attachments)

Detecting IRC Based Botnets

- Idea
 - Botnet communication often uses IRC
 - Botnet detector on top of IRC analyzer
 - Check nicknames
 - Check channel names
 - Check contact to identified bot-servers
- Key consideration: must *analyze* IRC dialog seen off-port
 - Because lots of benign IRC runs off-port too ...
- > 100 bots found at MWN+UCB
 - MWN employs **auto-blocking** based on detector

Performance Evaluation

		Stock- Bro	PIA- Bro	PIA- Bro-M4K
Config-A	Standard	3335s	3254s	
	Standard + sigs	3843s	3778s	
Config-B	All TCP pkts	3584s	3496s	
Config-C	All TCP pkts + sigs	4446s	4436s	3716s
	All TCP pkts + sigs + reass.		4488s	3795s

Performance

- New framework does not add significant additional overhead
- Protocol detection (signature matching on all packets) expensive but doable
 - Solutions:
 - Specialized hardware
 - Load balancing possible

Summary

- Network traffic resists classification by port
- General framework for dynamic protocol analysis
 - Use signatures to pre-filter for efficiency
 - Use application parsing to make high-quality decision
- Accurate enough for auto-blocking of bots at large-scale network
 - Plus detection of illicit relays & servers ...
- Integrated into upcoming Release 1.2 of Bro