BlindBox: Deep Packet Inspection over Encrypted Traffic

Justine Sherry, Chang Lan, Raluca Ada Popa, Sylvia Ratnasamy

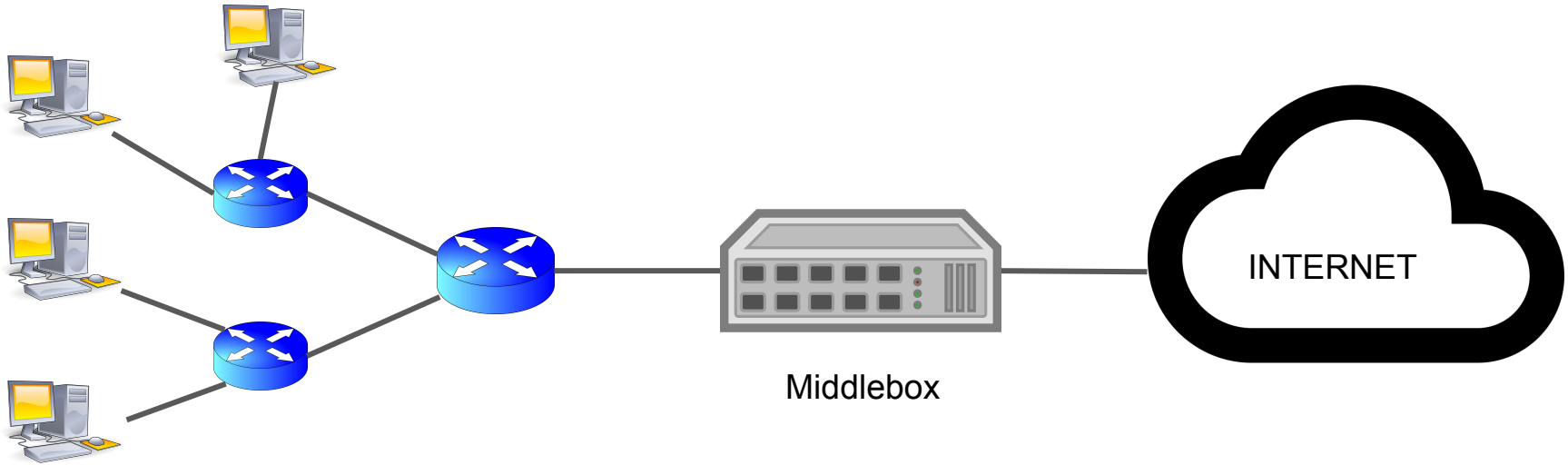
presentation by Luke Hsiao

Outline

- Introduction and Motivation
- System Overview
- Threat Model
- Functionality Evaluation
- Performance Evaluation
- Discussion

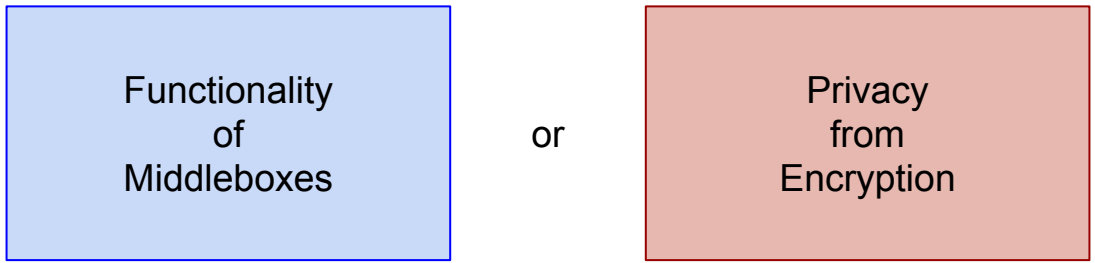
What is Deep Packet Inspection (DPI)?

- In-network middleboxes use DPI to examine and alter packets
- Used to enforce security policies
  - Intrusion Detection/Prevention, Exfiltration Prevention, Parental filtering, etc.



DPI and HTTPS

- HTTPS and other encryption protocols have dramatically grown in usage
- Packet payloads are encrypted, middleboxes can no longer inspect them
- To enable inspection, some systems support *insecure* HTTPS
  - Man-in-the-middle attack on SSL



Can we get both?

BlindBox: Both Privacy and DPI

- Performs inspection directly on *encrypted* payload
- Connection Setup:
  - sender/receiver bootstrap off SSL handshake
  - Middlebox performs own connection setup using obfuscated rule encryption
- Send:
  - Encrypts traffic with SSL, tokenizes traffic by splitting into substrings, encrypts tokens

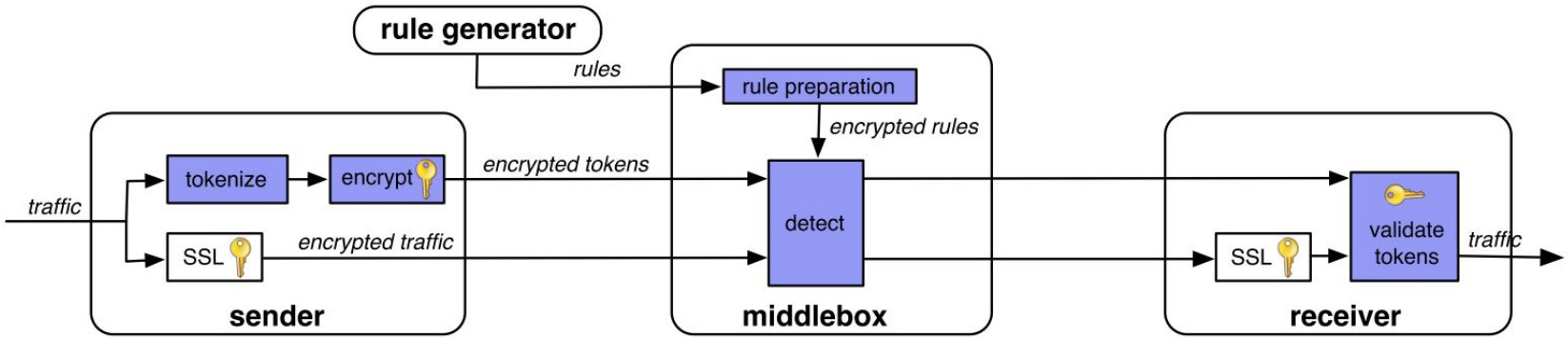


Figure 1: System architecture. Shaded boxes indicate algorithms added by BlindBox.

BlindBox: Both Privacy and DPI

- Detection
  - Middlebox receives both SSL-encrypted traffic and encrypted tokens
  - Detect module searches for matches between encrypted rules and encrypted tokens
- Receive
  - Receiver decrypts and authenticates traffic using normal SSL
  - Receiver also checks that encrypted tokens were encrypted properly by sender

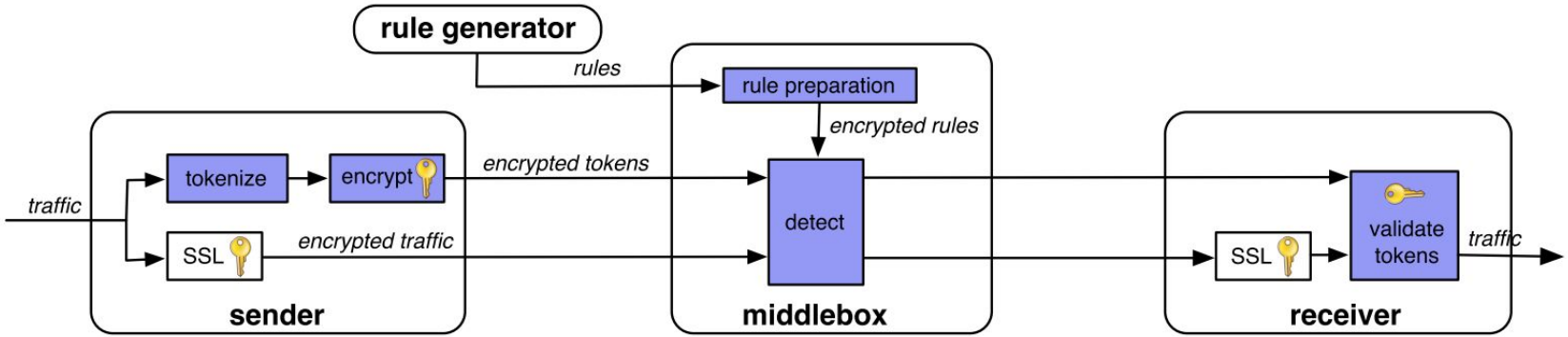


Figure 1: System architecture. Shaded boxes indicate algorithms added by BlindBox.

Threat Model Summary

- Clients
  - Want to protect privacy from middlebox AND protection from each other
  - Requires: at least one client must be honest
- Middlebox
  - Honest but curious
  - Can only see what is necessary to enforce security policy
- Rule Generator
  - Must be trusted by both middlebox and clients
  - Cannot actually observe or alter traffic

Functionality Evaluation

- Can BlindBox implement the functionality required for each target system?
  - Protocol I: Exact String Matching
    - Parental Filtering + Document watermarking
  - Protocol II: Exact String Matching for Multiple Keywords
    - Extends support to IDS policies requiring multiple keywords
  - Protocol III: Probable Cause Privacy
    - Supports RegEx and scripting, by enabling decryption w/ probable cause

Dataset	I.	II.	III.
Document watermarking [45]	100%	100%	100%
Parental filtering [13]	100%	100%	100%
Snort Community (HTTP)	3%	67%	100%
Snort Emerging Threats (HTTP)	1.6%	42%	100%
McAfee Stonesoft IDS	5%	40%	100%
Lastline	0%	29.1%	100%

Table 1: Fraction of attack rules in public and industrial rule sets addressable with Protocols I, II, and III.

Functionality Evaluation

- Does BlindBox fail to detect any attacks/policy violations that standard implementations would detect?
  - Environment: college “capture the flag” contest for hacking servers
  - Compared to Snort, BlindBox detected:

97.1% of attack keywords
99% of the attack rules

Performance Evaluation

- How long does it take to encrypt/detect a token?
- How long does the initial handshake take with the middlebox?
- How does BlindBox compare in detection time against other strawmen approaches?

		Vanilla HTTPS	FE Strawman	Searchable Strawman	BlindBox HTTPS
Client	*Encrypt* (128 bits)	13ns	70ms	2.7μs	69ns
	Encrypt (1500 bytes)	3μs	15s	257μs	90μs
	Setup (1 Keyword)	73ms	N/A	N/A	588 ms
	Setup (3K Rules)	73ms	N/A	N/A	97 s
MB	*Detection:*				
	1 Rule, 1 Token	NP	170ms	1.9μs	20ns
	1 Rule, 1 Packet	NP	36s	52μs	5μs
	3K Rules, 1 Token	NP	8.3 minutes	5.6ms	137ns
	3K Rules, 1 Packet	NP	5.7 days	157ms	33μs

Table 2: Connection and detection micro-benchmarks comparing Vanilla HTTPS, the functional encryption (FE) strawman, the searchable strawman, and BlindBox HTTPS. NP stands for not possible. The average rule includes three keywords.

Performance Evaluation

- How long are page downloads with BlindBox, excluding setup cost?
  - Single-core CPU can keep up with link rate

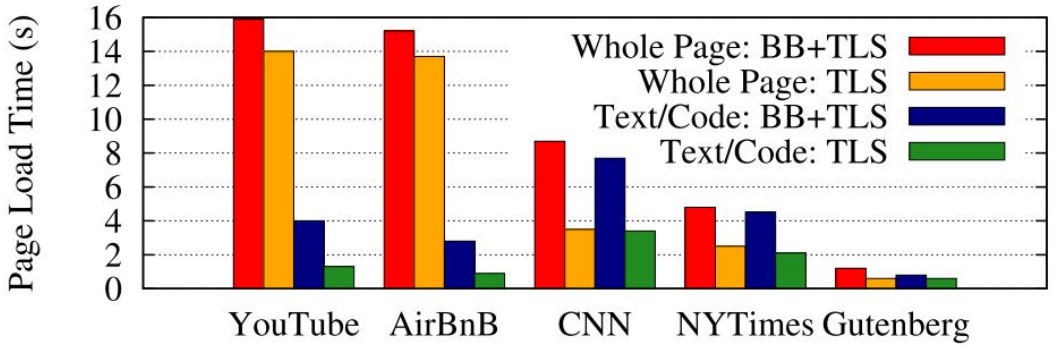


Figure 3: Download time for TLS and BlindBox (BB) + TLS at 20Mbps×10ms.

Performance Evaluation

- What is the computational overhead of BlindBox encryption, and how does it impact page load times?
  - Figure 4: Easy to see cost of encryption at a link capacity of 1Gbps
  - Can be mitigated with extra cores and parallelization

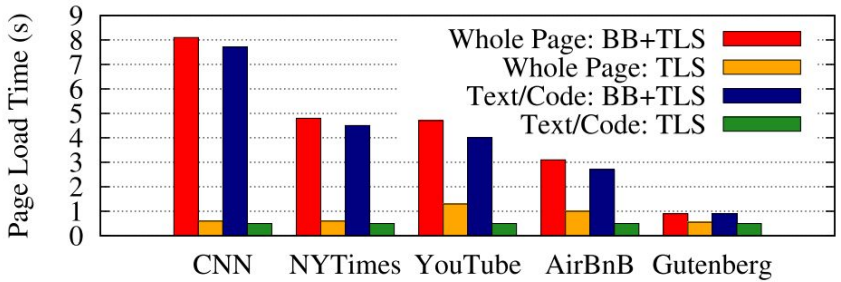


Figure 4: Download time time for TLS and BlindBox (BB) + TLS at 1Gbps×10ms.

Performance Evaluation

- What is the bandwidth overhead of transmitting encrypted tokens for a typical web page?
  - Depends on what fraction of bytes are text/code that must be tokenized
  - Penalty is lower for pages consisting mostly of video/images since BlindBox doesn't tokenize video/images.

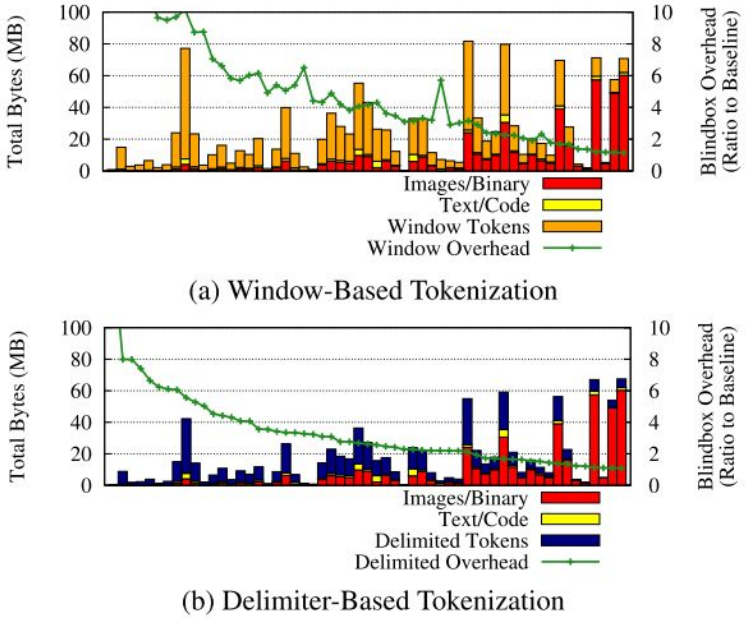


Figure 5: Bandwidth overhead over top-50 web dataset.

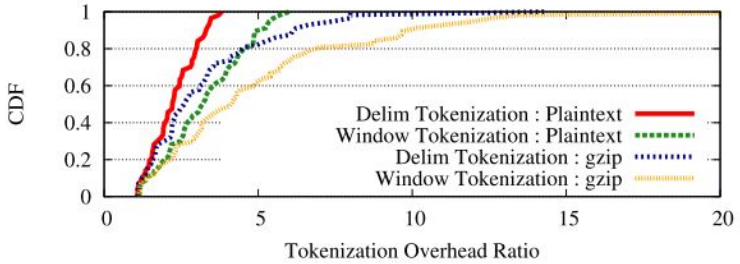


Figure 6: Ratio: transmitted bytes with BlindBox to transmitted bytes with SSL.

Evaluation Highlights

- Functionality:
  - Seems to cover the majority of use cases, esp. with protocol III
- Detection Time: similar to existing IDS
  - 186Mbps with BlindBox (compare to Snort at 85Mbps)
- Transmission Time: reasonable overhead
  - Page load completion time increases by 0.15-1x (ignoring setup)
- Setup Time: very slow
  - 97 sec for 3000 rules
  - This could be OK when connections are persistent

Discussion

- Alternatives to BlindBox?
  - Read-only middlebox protocol?
- Limitations of the threat model?
  - Can we always find a trusted rule generator?
  - Why must we keep rules hidden from endpoints?
    - Is it worth exposing rules to the ends in order to improve performance/reduce complexity?
  - Does decryption when matching a substring give MB too much power?
- Other applications of BlindBox?
  - IoT auditing? (Judson Wilson's work)
- How do we feel about their results?
  - Do we believe the numbers?
  - Are their metrics relevant measures of "success"?