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

![System architecture diagram](image)

*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

![System architecture diagram](image)

**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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>I.</th>
<th>II.</th>
<th>III.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document watermarking [45]</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Parental filtering [13]</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Snort Community (HTTP)</td>
<td>3%</td>
<td>67%</td>
<td>100%</td>
</tr>
<tr>
<td>Snort Emerging Threats (HTTP)</td>
<td>1.6%</td>
<td>42%</td>
<td>100%</td>
</tr>
<tr>
<td>McAfee Stonesoft IDS</td>
<td>5%</td>
<td>40%</td>
<td>100%</td>
</tr>
<tr>
<td>Lastline</td>
<td>0%</td>
<td>29.1%</td>
<td>100%</td>
</tr>
</tbody>
</table>

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?

<table>
<thead>
<tr>
<th>Client</th>
<th>Encrypt (128 bits)</th>
<th>Encrypt (1500 bytes)</th>
<th>Setup (1 Keyword)</th>
<th>Setup (3K Rules)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>13ns</td>
<td>3μs</td>
<td>73ms</td>
<td>73ms</td>
</tr>
<tr>
<td>FE Strawman</td>
<td>70ms</td>
<td>15s</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Searchable Strawman</td>
<td>2.7μs</td>
<td>257μs</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>BlindBox HTTPS</td>
<td>69ns</td>
<td>90μs</td>
<td>588 ms</td>
<td>97 s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MB</th>
<th>Detection:</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 Rule, 1 Token</td>
<td>NP</td>
<td>170ms</td>
<td>1.9μs</td>
</tr>
<tr>
<td></td>
<td>1 Rule, 1 Packet</td>
<td>NP</td>
<td>36s</td>
<td>52μs</td>
</tr>
<tr>
<td></td>
<td>3K Rules, 1 Token</td>
<td>NP</td>
<td>8.3 minutes</td>
<td>5.6ms</td>
</tr>
<tr>
<td></td>
<td>3K Rules, 1 Packet</td>
<td>NP</td>
<td>5.7 days</td>
<td>157ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

![Bar chart showing page load time for TLS and BlindBox (BB) + TLS at 20Mbps × 10ms.]

**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”?