Mitigating Active Attacks Towards Client Networks Using the Bitmap Filter

Chun-Ying Huang    Kuan-Ta Chen    Chin-Laung Lei

Distributed Computing and Network Security Lab
Department of Electrical Engineering
National Taiwan University

June 26, 2006
Outline

1. Introduction
2. The Bitmap Filter
3. Evaluations
4. Discussions
5. Conclusion
Outline

1 Introduction
2 The Bitmap Filter
3 Evaluations
4 Discussions
5 Conclusion
Active Attacks

Definition

An active attack is behavior that deliberately scans, probes, or intrudes on certain hosts or networks with malicious intent.
Active Attacks

**Definition**

An active attack is behavior that deliberately scans, probes, or intrudes on certain hosts or networks with malicious intent.

**Motivations**

- The popularity of Internet worms moves the victims.
- Most defense mechanisms are required to deploy globally.
- How does an ISP prevent customers/clients from attacks?
Active Attacks

Definition

An active attack is behavior that deliberately scans, probes, or intrudes on certain hosts or networks with malicious intent.

Motivations

- The popularity of Internet worms moves the victims.
- Most defense mechanisms are required to deploy globally.
- How does an ISP prevent customers/clients from attacks?
- Construct an efficient stateful packet inspection (SPI) filter.
Stateful Packet Inspection

Client – C

SPI Filter

Attacker – A

Server – S

Definitions and Motivations

Stateful Packet Inspection
Stateful Packet Inspection

State Table

<table>
<thead>
<tr>
<th>Src</th>
<th>Src-Port</th>
<th>Dst</th>
<th>Dst-Port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>C</td>
<td>1234</td>
</tr>
<tr>
<td></td>
<td></td>
<td>S</td>
<td>80</td>
</tr>
</tbody>
</table>

Request:
C:1234 → S:80

Client → SPI Filter → Server

Definitions and Motivations

Stateful Packet Inspection
Stateful Packet Inspection

State Table

<table>
<thead>
<tr>
<th>Src</th>
<th>Src-Port</th>
<th>Dst</th>
<th>Dst-Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>C</td>
<td>1234</td>
<td>S</td>
<td>80</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

Request:
C:1234 → S:80

Response:
S:80 → C:1234

Client - C

SPI Filter

Server - S

Attacker - A
Stateful Packet Inspection

State Table

<table>
<thead>
<tr>
<th>Src</th>
<th>Src-Port</th>
<th>Dst</th>
<th>Dst-Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1234</td>
<td>S</td>
<td>80</td>
</tr>
</tbody>
</table>

Client → C

SPI Filter

Request: C:1234 → S:80

Response: S:80 → C:1234

Attacker → A

Attack #1: A:80 → C:4567

Attack #2: X:1234 → C:5678
The Problem

The linearly increased costs on both storage spaces and computations.
Outline

1. Introduction
2. The Bitmap Filter
3. Evaluations
4. Discussions
5. Conclusion
Client Network Traffic Characteristics

Observations

1. Connection/Session lifetime
2. Out-In packet delay
## Client Network Traffic Characteristics

### Observations
1. Connection/Session lifetime
2. Out-In packet delay

### Data source: aggregated six class-C campus client networks
- A 6-hour TCP and UDP packet trace.
- Collected between 10AM and 4PM in a weekday.
- 96.25% are TCP packets; and 3.75% are UDP packets.
- Average packet rate: 24.63K packets per second.
- Average bandwidth utilization: 138.55 Mbps.
- Average packet size: 720 bytes.
Connection/Session Lifetime

**Definition**

Given a TCP connection, measure the time between the last TCP-SYN and the first TCP-FIN packet.

99% connections are shorter than 515 seconds.
Connection/Session Lifetime

Definition

Given a TCP connection, measure the time between the last TCP-SYN and the first TCP-FIN packet.

Result summary

- 90%: < 76 seconds.
- 95%: < 6 minutes.
- 99%: < 515 seconds.
- Lifetime is short.

99% connections are shorter than 515 seconds.
Definition of the Out-In Packet Delay

- A connection contains several outgoing and incoming packets.
- We measure the elapsed time between the outgoing packet and the successive incoming packets in each connection.
Result summary

- Observed port-reuse effect.
- 99% < 2.8 seconds, implies that Internet traffic is bi-directional and has high locality in the temporal domain.
Construct the Bitmap Filter

With the previous observations:

1. Connection/Session lifetime is short.
2. Out-in packet delays are short.
3. Internet traffic is bi-directional.

A stateful packet inspection (SPI) filter can be modified.
A Naïve Method to Modify an SPI Filter

Expire connection state information with timers.

---

State Table

<table>
<thead>
<tr>
<th>Connection Info.</th>
<th>Timer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>C:1234 → S:80</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Client → C

SPI Filter

Server → S

---

Request: C:1234 → S:80

Response: S:80 → C:1234

---

Attacker → A

Attack #1: A:80 → C:4567

Attack #2: X:1234 → C:5678

---

C.-Y. Huang, K.-T. Chen, C.-L. Lei

Mitigating Active Attacks Towards Client Networks
A Naïve Method to Modify an SPI Filter

Expire connection state information with timers.

<table>
<thead>
<tr>
<th>State Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection Info.</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>C:1234 → S:80</td>
</tr>
<tr>
<td>:</td>
</tr>
</tbody>
</table>

Howver: Still linear complexities on storages and computations.
Improved Performance: Using the Bitmap Filter

Reduce storages/computations complexities to constant.
Improved Performance: Using the Bitmap Filter

Reduce storages/computations complexities to constant.

Definition

- A **bitmap filter** is a composition of $k$ bloom filters of equal size $N$ ($=2^n$-bit), denoted as a $\{k \times N\}$-bitmap filter.
- The $i^{th}$ bloom filter is denoted as $\text{bit-vector}[i]$ in the algorithms.
Improved Performance: Using the Bitmap Filter

Reduce storages/computations complexities to \textit{constant}.

**Definition**

- A \textbf{bitmap filter} is a composition of $k$ bloom filters of equal size $N$ ($=2^n$-bit), denoted as a \{k × N\}-bitmap filter.

- The $i^{th}$ bloom filter is denoted as $bit-vector[i]$ in the algorithms.
The Algorithms

Initialization

- A \( \{k \times N\} \)-bitmap filter is initialized to all bits zero.
- All the \( k \) bloom filters are configured to share the same \( m \) hash functions.
The Algorithms

Initialization

- A \( \{k \times N\} \)-bitmap filter is initialized to all bits zero.
- All the \( k \) bloom filters are configured to share the same \( m \) hash functions.

The concept: Time-rotated bloom filters
The concept: Time-rotated bloom filters
The Algorithms

The concept: Time-rotated bloom filters

At time $= t_0$

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>...</th>
<th>K</th>
</tr>
</thead>
<tbody>
<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>

current

eldest

At time $= t_0 + \Delta t$

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>...</th>
<th>K</th>
</tr>
</thead>
<tbody>
<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>

eldest

current
The Algorithms

The concept: Time-rotated bloom filters

At time = $t_0$

```
1 2 3 ... K
... ...
... ...
... ...
```

current eldest

At time = $t_0 + \Delta t$

```
1 2 3 ... K
... ...
... ...
... ...
```

eldest current

At time = $t_0 + 2\Delta t$

```
1 2 3 ... K
... ...
... ...
... ...
```

current eldest
The Algorithms (cont’d)

Algorithm I: Periodically reset the eldest bloom filter

- Rotate one time unit, then reset the eldest bloom filter.
The Algorithms (cont’d)

**Algorithm I: Periodically reset the eldest bloom filter**
- Rotate one time unit, then reset the eldest bloom filter.

**Algorithm II: Test and set the bloom filters**
- Outgoing packets: Mark all corresponding bits on all bloom filters.
- Incoming packets: Reject if not all corresponding bits are marked on the current bloom filter.
The two algorithms implement the same concept as the modified SPI filter presented before.
The two algorithms implement the same concept as the modified SPI filter presented before.

At time = $t_k + \Delta t$, all columns are reduced by 1.
The two algorithms implement the same concept as the modified SPI filter presented before.

At time $= t_k + 2\Delta t$, all columns are reduced by 1, again.
The two algorithms implement the same concept as the modified SPI filter presented before.

At time $t_k + 2\Delta t$, with an occurrence of a new connection.
Parameter Decisions

Parameter List

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_e$</td>
<td>The expired time of a state.</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>The time unit to rotate the bitmap.</td>
</tr>
<tr>
<td>$k$</td>
<td>The number of used bloom filters.</td>
</tr>
<tr>
<td>$N$</td>
<td>The size of each bloom filter. The real size is $2^n$-bit.</td>
</tr>
<tr>
<td>$m$</td>
<td>The number of used hash functions for the bloom filters.</td>
</tr>
</tbody>
</table>

Guidelines

1. Recall the observed port-reuse effect. $T_e$ should not be too large.
2. $\Delta t$ should be set properly. Small $\Delta t$ increases system loads; large $\Delta t$ reduces system granularity (and precision).
3. $k$ is roughly $\lfloor \frac{T_e}{\Delta t} \rfloor$.
4. $N$ and $m$ depends on the scale of the network and the required precision.
False Positives and False Negatives

Definition

- *False positive*: Normal behavior is rejected.
- *False negative*: Attacks are accepted.
False Positives and False Negatives

Definition

- **False positive**: Normal behavior is rejected.
- **False negative**: Attacks are accepted.

False positives

- Since the lifetime of a state is $T_e$ seconds, a false positive occurs only when the out-in packet delay is longer than $T_e$ seconds.
- As the statistics show, when $T_e$ is greater than 2.8 seconds, the false positive rates should be lower than 1%.
False Negatives

Estimating on False Negative Rates

1. Given the expected max number of active connections $c$ in $Te$ and the bitmap size $N$, the false negative rate can be estimated by

$$p \leq \exp\left(-\frac{N}{e \cdot c}\right).$$  \hspace{1cm} (1)

2. In contrast, given $N$ and a tolerable maximum value of $p$, the expected max number of active connections $c$ should satisfy

$$c \leq -\frac{N}{e \ln p}.$$ \hspace{1cm} (2)
Examples

For a small- or medium-scale network

A bitmap filter is constructed using the following configuration

- Parameters: $k = 4$, $\Delta t = 5$, ($T_e = 20$), $m = 3$
- Required memory space: $\frac{k \times 2^n}{8} = 512$ K bytes.

Given the tolerable penetration probability of 10%, 5%, and 1%, the bitmap filter provides capacity of 167K, 125K, and 83K active connections in $T_e$, respectively.
For a small- or medium-scale network

A bitmap filter is constructed using the following configuration

- Parameters: $k = 4$, $\Delta t = 5$, ($T_e = 20$), $m = 3$
- Required memory space: $\frac{k \times 2^n}{8} = 512$ K bytes.

Given the tolerable penetration probability of 10%, 5%, and 1%, the bitmap filter provides capacity of 167K, 125K, and 83K active connections in $T_e$, respectively.

Compare with the campus network traffic

Only 15K active connections within a $T_e$ of 20 seconds.
### Summary

**Packet Processing:**
- For an outgoing packet: $O(m)$ hashes + $O(m \times k)$ marks.
- For an incoming packet: $O(m)$ hashes + $O(m)$ checks.

**Bitmap Rotation:**
- Reset a bit vector: constant according to the given bitmap size.
Performance

Summary

Packet Processing:

- For an outgoing packet: $O(m)$ hashes + $O(m \times k)$ marks.
- For an incoming packet: $O(m)$ hashes + $O(m)$ checks.

Bitmap Rotation:

- Reset a bit vector: constant according to the given bitmap size.

Since the bitmap is designed to have **fixed size and continuous memory space** and the components used in the algorithms are easy to implement as hardware, these algorithms can be completely implemented as a hardware easily.
## Compare with Similar Implementations

<table>
<thead>
<tr>
<th></th>
<th>Hash + link-list (Linux)</th>
<th>AVL-tree</th>
<th>Bitmap filter</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Storage space - Complexity</strong></td>
<td>O(n)</td>
<td>O(n)</td>
<td>O(c)</td>
</tr>
<tr>
<td><strong>Storage space - Handle 2.55M active connections</strong></td>
<td>77M bytes</td>
<td>77M bytes</td>
<td>8M bytes</td>
</tr>
<tr>
<td><strong>Computation Complexity - Insert a new state</strong></td>
<td>O(1)</td>
<td>O(log $n$)</td>
<td>O(1)</td>
</tr>
<tr>
<td><strong>Computation Complexity - Search for a state</strong></td>
<td>O(n)</td>
<td>O(log $n$)</td>
<td>O(1)</td>
</tr>
<tr>
<td><strong>Computation Complexity - Garbage collection</strong></td>
<td>O(n)</td>
<td>O(n)</td>
<td>O(c)</td>
</tr>
<tr>
<td><strong>Hardware acceleration</strong></td>
<td>Possible</td>
<td>Expensive</td>
<td>Cheap</td>
</tr>
</tbody>
</table>
Simulation I: Drop Rate Comparison

Compare the drop rate of BF and SPI-filter

Environments

- Implement an SPI filter (expire idle state after 240 seconds).
- The bitmap filter: \( n = 20, k = 4, \Delta t = 5, T_e = 20 \) (512K bytes).
- Drop rates measure in a 5-second time unit.
Simulation I: Drop Rate Comparison

Compare the drop rate of BF and SPI-filter

Environments

- Implement an SPI filter (expire idle state after 240 seconds).
- The bitmap filter: $n = 20$, $k = 4$, $\Delta t = 5$, $T_e = 20$ (512K bytes).
- Drop rates measure in a 5-second time unit.
Simulation II: Filter Rate

Environments

- The same bitmap filter used in simulation I.
- Attack rate: spoofed addresses, 500K packets per second.
Simulation II: Filter Rate

Environments

- The same bitmap filter used in simulation I.
- Attack rate: spoofed addresses, 500K packets per second.

### a. Filter Performance

<table>
<thead>
<tr>
<th>Time (in seconds)</th>
<th>Packet Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0e+00</td>
</tr>
<tr>
<td>5000</td>
<td>2e+05</td>
</tr>
<tr>
<td>10000</td>
<td>4e+05</td>
</tr>
<tr>
<td>15000</td>
<td>6e+05</td>
</tr>
</tbody>
</table>

### b. Attack Filtering Rate

<table>
<thead>
<tr>
<th>Time (in seconds)</th>
<th>Filtering rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12000</td>
<td>99.95</td>
</tr>
<tr>
<td>14000</td>
<td>99.97</td>
</tr>
<tr>
<td>16000</td>
<td>99.99</td>
</tr>
<tr>
<td>18000</td>
<td>100.01</td>
</tr>
</tbody>
</table>
Summary of Discussions

1 The Compatibility

- The bitmap filter is compatible with all single connection applications.
- For multiple connection applications, the bitmap filter is also compatible with hole-punching like NAT-traversal solutions.
Summary of Discussions

1. The Compatibility
   - The bitmap filter is compatible with all single connection applications.
   - For multiple connection applications, the bitmap filter is also compatible with hole-punching like NAT-traversal solutions.

2. Attack from Insiders
   - An inside attacker may quickly increase the bitmap utilization when attacking outsiders. It hence increase the false negative rate.
   - An administrator may increase $n$ or $T_e$ to tolerate such attacks. However, it would be better to identify and eliminate inside attackers.
Summary of Discussions

1. The Compatibility
   - The bitmap filter is compatible with all single connection applications.
   - For multiple connection applications, the bitmap filter is also compatible with hole-punching like NAT-traversal solutions.

2. Attack from Insiders
   - An inside attacker may quickly increase the bitmap utilization when attacking outsiders. It hence increase the false negative rate.
   - An administrator may increase $n$ or $T_e$ to tolerate such attacks. However, it would be better to identify and eliminate inside attackers.

3. Adaptive Packet Dropping
   - For bandwidth attacks only, the bitmap filter may consider to adopt adaptive packet dropping to increase the compatibility.
Outline

1. Introduction
2. The Bitmap Filter
3. Evaluations
4. Discussions
5. Conclusion
We propose the bitmap filter, an alternative implementation to replace the stateful packet inspection (SPI) filter, to stop malicious traffic for client networks.

The bitmap filter successfully reduces the complexities of both storage and computation to constants.

Analyses and simulations show that with limited resources, the bitmap filter can filter 90% even 99% attack traffic.
Thanks for your attention.
Comments or questions?