The FootFall Project
Tracing Attacks Through Non-Cooperative Networks and Stepping Stones with Timing-Based Watermarking

Douglas Reeves
Peng Ning
N.C. State University

Xinyuan Wang
George Mason University
The Problem

• You can’t defeat an attacker you can’t identify or locate
• The origin of network-based attacks can be very difficult to trace back
• There is little or no support for tracing from...
  – Internet protocols
  – equipment vendors
  – many network operators and service providers
Attack Origin Concealment Techniques

• Packet source addresses can be spoofed
• Attackers can use intermediate hosts (stepping stones, zombies, proxies) to attack their victim
  – Packet contents and headers can be encrypted or re-encrypted
Concealment Techniques (cont’d)

- Attacks may be slow, low volume
- Attack traffic can be mixed with innocuous traffic (other flows, chaff)
- Congestion may change traffic timing, and attackers may deliberately perturb traffic timing
Concealment Techniques (cont’d)

- Routers and proxies may alter packet counts (dropped / retransmitted packets, merging small packets into larger packets, …)
- Traffic flows may be split into subflows, merged again somewhere later…
Flow Watermarking

- Use traffic timing for tracing purposes
- Actively manipulate (watermark) traffic timing to aid identification
- Use redundant coding to overcome the effects of noise (normal or deliberate)
Project Objectives

• Be able to trace back to the origin of stepping stone attacks even if the attacker uses
  – Encryption (e.g., SSH)
  – Timing perturbation
  – Chaff
  – Drop/retransmit packets
  – Split and merge traffic flows
  – Anonymizing networks
  – Packet desynchronization
  – Short attacks (attacks with a small number of packets)
Technical Capabilities

• A number of watermarking schemes for trace back purposes
  – Encryption → timing perturbation → chaff → packet dropping and retransmission → anonymizing networks → split and merge → packet desynchronization → short attacks

• Analysis of the secrecy of watermarking

• Monitor deployment
  – Where and how many monitors we need to trace attackers
Project Status / Accomplishments

• Theoretical justification of claims
• Implemented in Linux kernel
• Tested, effectiveness demonstrated
• Delivered to sponsor, tested independently
• 5 papers published or to appear
  – CCS 03; SAC 05; SDCS 05; CCS 05; Oakland 06
  – Another 2 journal papers in submission
Recent Progress
(Since Last Joint PI meeting in October 05)

• Secrecy of watermarking
  – To be presented at Oakland 06

• New watermarking scheme to handle packet desynchronization
Secrecy of Watermarking

• If our manipulation of packet timing is easily detectable by attacker, can abort the attack immediately

• If the attacker can infer watermarking parameters, can…
  – remove (undo) watermark (degrade T.P. rate)
  – add same watermark to other flows (degrade F.P. rate)

• How resistant is our scheme to such inference?
Analysis of the Secrecy of Watermarking

- Analyze through examining the packet delays
  - How the existence of a watermark can be detected as soon as possible?
  - How to infer watermark parameters?
  - How to duplicate the watermark to normal flows to fool watermark decoders?

Attacker’s connections

Obtain delay for the same data items
A Hint

- Packet delays with and without watermark
  - Normal network delay: normal distribution
  - Watermark delay: uniform distribution over $[0, S]$
  - Packet delay with watermark: Normal + uniform

Challenges: (1) Not all packets are delayed by WM; (2) $S$ is unknown!
Summary of the Results on WM Secrecy

- Can quickly determine whether a flow is being watermarked
  - (Modified) Sequential Probability Ratio Test (SPRT)
- Can recover or duplicate WM in some cases
  - Cases I & II: *same packet reused*
    - Can recover the WM parameters with high accuracy
  - In Cases III & IV: *packets forming IPDs not interleaved*
    - Can duplicate WM in other flows with high probability
  - Case V: others
    - Cannot recover or duplicate WM parameters
Experiment Results

- Estimation of quantization step $S$

<table>
<thead>
<tr>
<th>$S$</th>
<th>400ms</th>
<th>600ms</th>
<th>800ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated $S$</td>
<td>389.1ms</td>
<td>587.8ms</td>
<td>782.9ms</td>
</tr>
</tbody>
</table>

- Watermark recovery rate for case I
  - $L$: the number of watermark bits

<table>
<thead>
<tr>
<th>$L$</th>
<th>16</th>
<th>24</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$=400ms</td>
<td>97.23%</td>
<td>96.79%</td>
<td>96.70%</td>
</tr>
<tr>
<td>$S$=600ms</td>
<td>97.12%</td>
<td>97.29%</td>
<td>97.05%</td>
</tr>
<tr>
<td>$S$=800ms</td>
<td>97.12%</td>
<td>97.33%</td>
<td>97.40%</td>
</tr>
</tbody>
</table>
Experiment Results (Cont’d)

- Watermark recovery rate for case II
- Watermark duplication rate for case III
  - \( K \): one of every \( K \) packets is used to embed watermark

<table>
<thead>
<tr>
<th></th>
<th>L</th>
<th>16</th>
<th>24</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>S=400 ms</td>
<td>93.17%</td>
<td>93.26%</td>
<td>94.10%</td>
<td></td>
</tr>
<tr>
<td>S=600 ms</td>
<td>93.23%</td>
<td>94.21%</td>
<td>93.55%</td>
<td></td>
</tr>
<tr>
<td>S=800 ms</td>
<td>93.41%</td>
<td>93.17%</td>
<td>94.60%</td>
<td></td>
</tr>
</tbody>
</table>

![Graph showing Watermark duplication rate (M=1) vs K (1 embedding packet is selected every K packets)]
Experiment Results (Cont’d)

- Watermark duplication rate for case IV
  - $M = 4$ (a watermark bit is embedded on 4 IPDs)

- Watermark removal rate for case IV
  - $M = 8$
Experiment Results (Cont’d)

- **Watermark detection rate**
  - Original SPRT: stop as soon as if finds no watermark at the beginning of the flow
  - Modified SPRT: keep checking for the entire flow

- **The number of packets processed before SPRT detect the watermark**

<table>
<thead>
<tr>
<th>K</th>
<th>2</th>
<th>4</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>M=1</td>
<td>100%</td>
<td>100%</td>
<td>99.7%</td>
</tr>
<tr>
<td>M=2</td>
<td>100%</td>
<td>98.9%</td>
<td>0%</td>
</tr>
<tr>
<td>M=3</td>
<td>98.6%</td>
<td>98.4%</td>
<td>0%</td>
</tr>
<tr>
<td>M=4</td>
<td>98.6%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

- **Modified SPRT**: keep checking for the entire flow
- 100% detection rate in all cases
Handling Packet Desynchronization

- Repacketization at application-layer (SSH)
  - Due to active timing perturbations, network delays, etc.

  **Incoming packets:**
  1 2 3 4 5 6 7 8 9 ...  
  **Outgoing packets:**
  1 2 3 4 5 6 7 8 9 ...  
  **IPD1**  
  **IPD2**  
  **desynchronized packets**

- Packet "desynchronization" at decoder
Interval-Based Watermarking

- Relaxed synchronization using “fixed intervals”

- Packet count differences of interval pairs

- Distribution of packet count differences
  - Symmetric
  - Sample mean converges to 0 as sample size grows
Single Watermark Bit Encoding

• Encoding of bit ‘0’:

  Watermark Embedding Intervals

  Difference = 0

  Difference = 2

• Encoding of bit ‘1’:

  Watermark Embedding Intervals

  Difference = 0

  Difference = 2
Detection Rate without Timing Perturbation

- Repacketization Rate = 0.002
- False Positive Rate < 1.E-05
- Detection Rate = 100% with about 1000 packets and h=5
Detection Rate with Timing Perturbation

- Repacketization Rate: 0.002~ 0.06 (proportional to perturbation)
- Highly robust against plausible timing perturbations
What’s Next

• Pushing the boundaries
  – Refine the new watermarking scheme
  – Split and merging
  – More on anonymizing networks
  – Non-interactive attacks
  – Short attacks
Adopters

• NSA?
  – Have done extensive test of our software

• Others
  – Trying our software is always welcome
Tracing Attacks with Timing-Based Watermarking
D. Reeves, P. Ning, X. Wang

What are you trying to do?
- Trace attack traffic quickly despite encryption, stepping stones, timing modifications, translations,…
- No modifications required to routers

Milestones/Schedule

<table>
<thead>
<tr>
<th>Task</th>
<th>FY06 Q1</th>
<th>FY06 Q2</th>
<th>FY06 Q3</th>
<th>FY06 Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proof of concept</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detector placement</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control “console”</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defeats chaff</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internet demo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defeat repacketization</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimal watermark</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defeat split / merge</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Progress Since Last Review
- Now can handle repacketization of the flow (SSH!)
- New results on security of method (robustness against protocol analysis)
  - accepted for presentation at IEEE Security and Privacy, 2006
Executive Summary

• A tracing method effective against
  – source address spoofing / translation, packet encryption, stepping stones / zombies, stealthy attacks, traffic mixing, timing perturbations, packet dropping / retransmission, split and merge, packet desynchronization, …

• Best of show (overall)
  ✔ generality (only restriction: enough packets)
  ✔ robustness (can be made arbitrarily effective)
  ✔ speed (requires fewest # of packets)

• Costs: must modify flow timing, deploy monitors
Thank You!

Questions or Comments?