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

Verona, NY
October 25, 2005
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

1. Packet source addresses can be spoofed

2. Packet contents and headers can be encrypted or re-encrypted

3. Attackers can use intermediate hosts (stepping stones, zombies, proxies) to attack their victim
Concealment Techniques (cont’d)

4. Attacks may be slow, low volume

5. Attack traffic can be mixed with innocuous traffic (other flows, chaff)

6. Congestion may change traffic timing, and attackers may deliberately perturb traffic timing
7. Routers and proxies may alter packet counts (dropped / retransmitted packets, merging small packets into larger packets, …)

8. Traffic flows may be split into subflows, merged again somewhere later…
Flow Watermarking

I. Use traffic **timing** for tracing purposes

II. Actively manipulate (**watermark**) traffic timing to aid identification

III. Use **redundant coding** to overcome the effects of noise (normal or deliberate)

**packets in a flow**

```
| ipd1 | ipd2 | ipd3 |
```

```
time →
```
Benefits of Our Approach

- Effective against source address spoofing / translation, packet encryption, stepping stones / zombies, stealthy attacks, traffic mixing, timing perturbations, packet dropping / merging, ...

- 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
Project Status / Accomplishments

- Theoretical justification of claims
- Implemented in Linux kernel
- Tested, effectiveness demonstrated
- Delivered to sponsor, tested independently
- 4 papers published, 4 submitted currently
- Now: pushing the boundaries
Recent Progress
#1: Defeating Chaff

- To disguise attacker traffic, can be mixed with…
  - padding traffic or chaff
  - other legitimate traffic flows

- Our basic watermarking scheme depends on the accuracy of packet counting
Chaff: Approach

- At a stepping stone, for each outgoing flow...
  - identify possible matching packets for each incoming packet
  - decode watermark based on closest match
  - use closest matching flow

- Assumptions
  - no packet loss or repacketization
  - packets are not reordered
Chaff: Results

- Good true positive rate
- Low false positive rate
- Medium computational cost
- Unrealistic assumptions 😞

Figure 5: Computation costs changing with $\lambda_c$, $\Delta = 7s$, correlated flows
Chaff: Flow Split and Merge

If a flow is split, one piece is watermarked, and later the subflows are merged back together, the non-watermarked subflows are chaff!

We applied our technique on this case {0% false positive rate}.
A necessary component of our approach is the use of monitors to detect the presence of a WM.

- Monitors must be placed close to all networks that could be the origin of an attack.

Q1.: What’s the fewest number of monitors needed to trace the origin of an attack with high probability?

Q2.: Where should these monitors be placed?
Graph Separators

- Known problem with known solutions: given graph $G$, find a separator of a specific size which partitions $G$, such that either the minimum or the average partition size is minimized.

<table>
<thead>
<tr>
<th>Graph</th>
<th>$\equiv$</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G$</td>
<td>Internet AS Topology</td>
<td></td>
</tr>
<tr>
<td>Separator</td>
<td>Set of monitors deployed</td>
<td></td>
</tr>
<tr>
<td>Separator size</td>
<td>Number of monitors</td>
<td></td>
</tr>
<tr>
<td>Partition</td>
<td>Set of non-monitored ASes connected by non-monitored links</td>
<td></td>
</tr>
<tr>
<td>Partition size</td>
<td>Ambiguity of attack attribution</td>
<td></td>
</tr>
</tbody>
</table>
Graph Partitioning for Monitors

Non-monitor
Monitor
Attacker
Victim
Separator Variations

- Vertex separator: one monitor per vertex (AS)
- Edge separator: one monitor per edge (link)
- Hybrid: one monitor per Internet Exchange (peering point)
- Weighted separator: vertexes are weighted, optimize weighted ambiguity
  - Ex.: weight = probability of being attack origin
- Separating the routing tree rooted at a specific network of interest
AS-level Internet Topology

- AS peering exhibits power law distribution

Degree Histogram of ASes on AS-level Internet Topology
(12538 ASes, 32932 Links / Skitter 8/2003)
Vertex Separator: Priority Based

- With 5% of ASes monitored, the maximum ambiguity is 2.7 and the average is 1.1
Edge Separator: Priority Based

- With 15% of links monitored, average ambiguity is 2.0, but maximum is over 1000
Monitor Deployment: Conclusions

- Attack origin isolation is…
  - cheap if defending single network
  - cheap if ASes can be easily monitored
  - expensive if links must be monitored, any AS can be target, any AS can be source, with equal probabilities

- Another conclusion: disruption of 3 key ASes would separate Internet into partitions of average size 10
#3: Security of Watermarking

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

2. 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?
Detecting Watermarks

- If you have both unwatermarked and watermarked versions of flow, are delays from
  - normal network delays? or,
  - watermarking?

- If you know distributions of normal network delays and watermarking delays, test if enough packets have the characteristics of watermarking
  - using the Probability Ratio Test
Detecting Watermarks (cont’d)

- Network delays normally distributed, watermarking delays uniformly distributed

<table>
<thead>
<tr>
<th>Ratio →</th>
<th>1.0</th>
<th>.95</th>
<th>.90</th>
<th>.85</th>
<th>.80</th>
<th>.75</th>
<th>.70</th>
<th>.65</th>
<th>.60</th>
<th>.55</th>
<th>.50</th>
</tr>
</thead>
<tbody>
<tr>
<td>WM delay</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>=200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>=300</td>
<td>1</td>
<td>1</td>
<td>.05</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>=1000</td>
<td>1</td>
<td>1</td>
<td>.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Test: ratio of > .94 = no WM, < .85 = WM
Inferring WM Parameters

- First parameter: quantization step size $s$

- Approach: Expectation-Maximization algorithm
  1. Guess an initial value for $s$
  2. Iteratively update the value of $s$
     - $E$-step: compute expected values
     - $M$-step: improve the likelihood
Inferring Which IPDs are Watermarked

- Select a **threshold** $h_d$
  - if packet delay $d_j \geq h_d \Rightarrow$ embedding packet
  - use Bayes’ decision rule to minimize the risk of errors

- If embedding packets are known, we can
  - recover watermark value, or
  - duplicate watermark value
Ability to Duplicate the WM

Without our estimation technique, random duplications would yield a 50% successful duplication rate.
Security Conclusions

- If watermarked and unwatermarked flows both available, possible to guess watermark parameters with reasonable accuracy.

- Goal: improve the security of the watermark.
#4: Packet Dropping and Retransmission

- If a host retransmits a dropped packet, sender will have one more packet than receiver.

- Solution: eliminate retransmitted (duplicate) packets from watermarking (encode and decode).

- Cost: watermarking module must maintain list or vector of received / transmitted packets.
#5: Control Console

- Source and destination IP addresses are entered in the interface.
- A configuration file path is specified.
- The Netfilter hook is set to 'FORWARD'.
- Encoding and execution options are available.

Communication currently unencrypted.
Conclusions
Current Focus: Wide Scale Testing

- Using hosts around the world as stepping stones, based on PlanetLab testbed
  - real Internet traffic
  - real (heavily-loaded) hosts
  - real applications

- First problem encountered: repacketization
Current Focus: Repacketization

- Repacketization interferes with WM detection

   ![Diagram showing incoming and outgoing packets with some packets marked in purple]

- Proposed solution: watermark timing based on byte counts, not (only) packet counts
  - not affected by repacketization

- Challenge: choose right byte count boundaries without knowing (a) size of packets (b) transmission rate of the traffic source
Current Focus: Compressed WMs

- There are known bounds on the information capacity of a side channel, such as packet timing
- We aren’t close to those bounds
- Plan: use a sophisticated compression technique to reduce the number of packets needed
Current Focus: Optimum Watermarking

- Lots of “knobs” to twiddle – what setting is best for…
  - a specific type / rate of traffic?
  - a specific set of attacker countermeasures?

- There are some easy improvements achievable with low effort
Phase II Goals

A. Broader, more stressful field testing

B. Tougher challenges: repacketization, loss of synchronization, watermark restoration, chaff, flow splitting / merging

C. Support for “customer evaluation”

D. Fewer packets + measure effectiveness on anonymizing systems

E. Hardening / integration / packaging

F. High-speed execution
Questions or Comments?