Outline

- Problem and Assumptions
- Sketch of the Method
- Results
- Recent Work
- Tech Transfer
The Problem

- Identify the true origin of Internet traffic
The Problem

Internet

Attacker?! Stepping Stone

Attacker?!

Stepping Stone

Victim
The Problem

- Packet headers removed / replaced
- Packet payloads (re-)encrypted
Metrics

- **Scenario:** “As shown on network TV”
- **Effective** (always finds source, never falsely identifies a non-source, very little traffic required)
- **Cheap**, simple to implement
- **Scales** to high-speed network performance
Metrics

- **Specific** (identifies IP address, or geographic location, of source)
- **General-purpose** (any type of traffic)
- **Unassisted** (no support from network operators, no network visibility)
- **Robust** (undetectable by adversary, impossible to evade)
Assumptions About Adversary

- Uses every evasion technique possible, and knows our method
  - **Basic**: address spoofing
  - **Application layer**: payload encryption, header modification
  - **Network**: repacketization of traffic
  - **Traffic modifications**: padding traffic, multi-path routing
  - **Timing jitter**: accidental (network) or intentional (by adversary)
The Basic Idea

- How do you trace digital data? You **watermark** it
  - Stealthy, robust, cheap, ...

- Applications
  - Video
  - Audio
  - Network traffic?
The Basic Idea

- We propose to modify (watermark) the traffic timing (spacing of packets)
  - Timing = a side channel
- Small changes, redundant coding
- Novelty: active manipulation of traffic timing ➔ major advantages
The Basic Idea

- Watermark parameters
- Watermark timing pattern to embed
- Victim
- Watermark generator
- Analyst
- Embedder
- Detector
- AS
- Attacker
- "I'm under attack!"
- "watermark detected"
- "watermark detected"
- "attack traffic (bidirectional)"

Here's what to watch for:
Basic Idea

Watermark = “This is suspicious traffic flow #235”

Input Traffic Flow (carrier)

Embedder

Encode

Modulate

Traffic Flow with Modified Timing

Internet

Traffic Flow with Modified Timing

Detector

De-modulate

Decode

Destination

Watermark Detected: “This is suspicious traffic flow #235”
Results

- Theoretically: as robust as you want to make it (just use more packets)

- In practice: implemented (PC+Linux), tests in…
  - Our lab
  - Independent lab
  - The global Internet
Results

- Using several hundreds of packets...
  - 98%+ detection rate
  - $10^{-4}$ false alarm rate

- Modulation is a few 100 ms. of jitter ($S$)

- Robust to 5-10 s. adversarial jitter ($N$)

- Robust to 20-50% padding (chaff packets)
A Test

- Four stepping stones (heavily loaded), at least 75 routers, in the Internet
Results

- Delay jitter $< 100\text{ms}$ more than 95% of the time
Scoresheet

- **Effective** (✅ always finds source, ✅ never falsely identifies a non-source, 😞 very little traffic required)

- ✅ Cheap, 😞 simple to implement

- ✗ Scales to high-speed network performance
Scoresheet

- **Specific** (★ identifies IP address, or ✗ geographic location, of source)

- **General-purpose** (any type of traffic)

- **Unassisted** (★ no support from network operators, ✗ no network monitors needed)

- **Robust** (😊 undetectable by adversary, ✔ impossible to evade)
Scoresheet: Evasion Techniques

- **Basic**: ⚫ address spoofing

- **Application layer**: ⚫ payload encryption, ⚫ header modification

- **Network**: 😞 repacketization of traffic

- **Traffic modifications**: 😞 padding traffic, ❌ multi-path routing

- **Timing jitter**: ⚫ accidental (network) or ✔ intentional (by adversary)
Recent Work: Repacketization

- TCP repacketization by stepping stones is a fact of life
  - Due to active timing perturbations, network delays, etc.

Incoming packets:
- 1 ➔ 2 ➔ 3 ➔ 4 ➔ 5

Outgoing packets:
- 1 ➔ 2 ➔ 3 ➔ 4 ➔ 5 ➔ 6

IPD1: [1 2 3 4 5]
IPD2: [6 7 8 9]

Desynchronized packets: 4, 5
Repacketization Results

- A new block-based watermarking method tolerates typical (< 15%) repacketization rates
Resynchronization Results

- New method is very robust to loss of synchronization, resynchronizes automatically
Recent: Detector Placement
Detector Placement Results

- The **good news**: we have a close-to-optimal algorithm (minimum number of detectors), and it scales to Internet size.

- The **bad news**: you’re going to need a lot of detectors for **accurate** localization.
Ex.: Detector Placement Results
Recent Work: Adaptive WM

- Idea: use native traffic characteristics to optimize the watermarking

- Results
  - 50-80% fewer packets needed, or
  - 70-90% smaller timing adjustment

- Requires embedder / detector real-time coordination
Recent Work: “Breaking” Our Method

- What is the likelihood the adversary can detect / decode our watermark, and…
  - Duplicate it?
  - Remove it?

- Insight: unmodified and modified timing can be distinguished

![Graph showing Normal and Normal + Uniform distributions.](image-url)
Results: Breaking Our Method

- Can quickly determine whether a flow is being watermarked
  - using (Modified) Sequential Probability Ratio Test (SPRT)

- Have shown how to add additional randomness to our method to defeat such analysis
Summary

- 😊 Most powerful, robust method for traffic tracing
  - We’ve defeated or neutralized pretty much every evasive technique we know
  - Xinyuan Wang: application to VoIP and anonymization

- 😞 Requires deployment of watermark embedders and detectors
What’s Left?

- The Grand, Unified Theory of Traffic Watermarking
  - The noise model is not Gaussian 😞

- “At-speed” execution
Tech Transfer

- **Papers**: CCS2003, CCS2005, S&P2006, + several workshops + journal submissions

- **Students**: 2 graduated, 2 in progress

- **Software prototype**, provided to technical contract monitor for evaluation

- PI cleared, can brief and support testing
Tracing Attacks Through Non-Cooperative Networks and Stepping Stones with Timing-Based Watermarking

D. Reeves, P. Ning, X. Wang

**Objectives**
- Trace attack traffic *quickly* despite payload encryption, use of stepping stones, traffic timing modifications, etc.
- Not require modification to routers

**Features**
- Embed identifying “watermark” in *covert timing channel*, and use to correlate flows
- Can be made *arbitrarily resistant* to attempts to modify traffic timing and remove watermark
- Difficult for attacker to detect, requires low number (few hundreds) of packets

**Schedule**

<table>
<thead>
<tr>
<th>Status</th>
<th>Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>done</td>
<td>Highly robust to timing perturbations</td>
</tr>
<tr>
<td>done</td>
<td>Optimal placement of watermark detectors</td>
</tr>
<tr>
<td>prototype</td>
<td>Control and analysis console</td>
</tr>
<tr>
<td>done</td>
<td>Internet-scale test and demonstration</td>
</tr>
<tr>
<td>done</td>
<td>Works despite traffic padding</td>
</tr>
<tr>
<td>done</td>
<td>Resists attempts to analyze, remove watermark</td>
</tr>
<tr>
<td>done</td>
<td>Independent of packet counting (resists packet dropping, repacketization)</td>
</tr>
<tr>
<td>done</td>
<td>Optimal detector placement</td>
</tr>
<tr>
<td>done</td>
<td>Optimal, adaptive watermark design</td>
</tr>
<tr>
<td>in prog.</td>
<td>Resynchronization of WM detector</td>
</tr>
<tr>
<td>N.A.</td>
<td>At-speed execution, scalability demonstration</td>
</tr>
</tbody>
</table>
Executive Summary - 1

- Attacks are staged through a sequence of hosts; difficult to identify true source

- Attackers use many techniques to avoid tracing
  - Address spoofing, stepping stones, padding traffic, anonymizers, timing modification, traffic encryption, repacketization, flow split/merge, etc…
We propose to modify (watermark) the traffic timing (spacing of packets)

- Timing = a side channel
- Stealthy, robust, cheap,

Small changes in timing, redundant coding to resist attacker/network noise injection (jitter)
Executive Summary - 3

- Modulation adds 10-100 ms. of jitter ($S$). Using a few 100’s of packets…
  - 98%+ detection rate, .01% false alarm rate
- Robust to: use of stepping stones, 5-10 sec. of adversarial jitter ($N$), 20-50% padding (chaff packets), 10% repacketization
- No assumptions about attack traffic type
- Undetectable by adversary
- Requires deployment of watermark embedders near protected assets, and detectors near suspected attack source
- Software prototype, available on request (suitable only for low-bandwidth analysis at present)
Watermark consists of a sequence of 1 bits

Decoding one bit: compute inter-packet delay (IPD) modulo a fixed step size s

Encoding algorithm...
- 0: increase IPD so modulo s = 0
- 1: increase IPD so modulo s = .5s

Another important parameter: packet selection function f()
Decoding 1 bit: compare the average IPDs of two groups of packets

Encoding algorithm…
- 0: increase IPDs of group A, decrease IPDs of group B, so group A avg IPD > group B avg IPD (probably)
- 1: increase IPDs of group B, decrease IPDs of group A, so group A avg IPD < group B avg IPD (probably)

Important parameters
- amount of timing increase \( a \)
- \# of packets per group + packet selection function \( f() \)
extra slide: A Test Case

The “stop-light” timing perturbation model
Methods 1, 2: WM Detection Rates

True Positive Rate Comparison (with FS1 of 97 Real Flows) between Quantization WM and Probabilistic WM under Uniform Perturbation

Higher is better
Methods 1,2: False Positive Rates

Lower is better
Passive methods have advantages (undetectable, simple); any drawbacks?

- 10-1000x more packets needed than watermarking

Other restrictions / assumptions
extra slide: Dealing with Padding (Chaff)

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

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

- Good true positive rate
- Low-ish false positive rate
- Medium computational cost
- Unrealistic assumptions
**Method 3: Interval Packet Counts(!)**

- **Encoding of bit ‘0’:**
  - Difference = 2
  - Difference = 0
  - Watermark Embedding Intervals

- **Encoding of bit ‘1’:**
  - Difference = 2
  - Difference = 0
  - Watermark Embedding Intervals
Detector Placement: Example

Non-monitor
Monitor
Attacker
Victim