Forensic Investigation of the OneSwarm Anonymous FileSharing System

Swagatika Prusty
Brian Neil Levine    Marc Liberatore

http://forensics.umass.edu

Dept. of Computer Science
UMass Amherst

This work was supported in part by NSF award CNS-1018615
Forensic Investigation and P2P Systems

**Why investigate?**

- Primary Use of P2P Systems: File Sharing

- **Social and legal concern** when used for sharing files containing child pornography (CP).

- **Goal** of forensic investigation is to acquire enough evidence to support a warrant.
  
  - To acquire evidence, meeting “**probable cause**” standards that an IP address is associated with possession or distribution of CP.
“Probable Cause” for Search Warrant

- The standard by which an officer or agent of the law has grounds to obtain a search warrant.

- P2P communications observed by law enforcement are in “plain view” and not protected by the 4th Amendment.

- This plain view evidence becomes the probable cause that permits search and seizure at a physical location.

- "Probable cause" is not quantitatively defined.
  - “a fair probability”
State of the Art

- Law Enforcement tries to identify and gather evidence against peers responsible for distribution of CP.

- Gnutella and other P2P networks are actively monitored by law enforcement:
  - Investigators join P2P network as ordinary peers.
  - They query for CP files.
  - Get replies directly from peers that possess these files.
  - This reveals the IP address of peers sharing CP.
  - With a subpoena, they track down a physical location.
  - Finally, a warrant is issued now that the “probable cause” standard is met.
The Challenge

- Investigations are thwarted in anonymity systems.
- Network identity of query originators and data sources is obscured.

- **OneSwarm** is one such P2P system designed with the goal of privacy preservation.
  - [Isdal et al., ACM SIGCOMM 2010]
  - An attractive venue for trafficking in child pornography files.

- Can plain view evidence sufficient for probable cause be acquired from OneSwarm?
Our Contributions

- We developed techniques for identification of peers sharing child pornography in the OneSwarm network.
  - Our techniques work within the constraints of criminal procedure.
  - In general, our techniques allows anybody to identify sources of files of interest.
    - Civil investigations (MPAA) for copyright infringement have a lower standard to meet than we do here.
  - We prove that OneSwarm does not sufficiently provide anonymity against third party monitoring.
Brief Overview of OneSwarm

- It is an anonymous P2P file sharing system.

Message Routing
- Peers search for files by sending keyword/infohash queries.
- Queries for files are **forwarded** until sources are found.
- Reply travels along the **reverse path** of search query.

Trust Relations
- A peer can categorize its neighbors as either “**trusted friends**” or “**untrusted friends**”.
- A peer subscribes to a **community server** that assigns links to up to 39 untrusted friends.
- A peer can add its real-world friends as “trusted friends” on OneSwarm by out-of-band methods.
  - Google talk import, LAN import or manual public key exchange.
Probabilistic Query Forwarding

1. A peer sends a query to its neighbors.
2. If the query receiving peer has the requested file, it replies back.
3. Else, it forwards the query to its neighbors with probability $p$.
4. This thwarts Collusion Attack.
OneSwarm: Anonymity Mechanisms

Probabilistic Query Forwarding

1. A peer sends a query to its neighbors.
2. If the query receiving peer has the requested file, it replies back.
3. Else, it forwards the query to its neighbors with probability $p$.
4. This thwarts Collusion Attack.
OneSwarm: Anonymity Mechanisms

Probabilistic Query Forwarding

A peer sends a query to its neighbors.

If the query receiving peer has the requested file, it replies back.

Else, it forwards the query to its neighbors with probability \( p \).

This thwarts Collusion Attack.
OneSwarm: Anonymity Mechanisms

Untraceable Queries

1. Queries do not contain a Time-Time-Live field.
   • Avoids traceback and attribution.
2. **Search Cancel** messages are introduced.
3. Queries are forwarded only after a delay of 150ms.
4. Cancel messages aren’t delayed.
5. This delay allows “search cancel” messages to catch up, mitigating flooding.
OneSwarm: Anonymity Mechanisms

Untraceable Queries

1. Queries do not contain a Time-Time-Live field.
   • Avoids traceback and attribution.
2. **Search Cancel** messages are introduced.
3. Queries are forwarded only after a delay of 150ms.
4. Cancel messages aren’t delayed.
5. This delay allows “search cancel” messages to catch up, mitigating flooding.
OneSwarm: Anonymity Mechanisms

Untraceable Queries

1. Queries do not contain a Time-Time-Live field.
   • Avoids traceback and attribution.
2. Search Cancel messages are introduced.
3. Queries are forwarded only after a delay of 150ms.
4. Cancel messages aren’t delayed.
5. This delay allows “search cancel” messages to catch up, mitigating flooding
OneSwarm: Anonymity Mechanisms

Untraceable Queries

1. Queries do not contain a Time-Time-Live field.
   • Avoids traceback and attribution.

2. **Search Cancel** messages are introduced.

3. Queries are forwarded only after a delay of 150ms.

4. Cancel messages aren’t delayed.

5. This delay allows “search cancel” messages to catch up, mitigating flooding.
OneSwarm: Anonymity Mechanisms

Untraceable Queries

1. Queries do not contain a Time-Time-Live field.
   - Avoids traceback and attribution.

2. **Search Cancel** messages are introduced.

3. Queries are forwarded only after a delay of 150ms.

4. Cancel messages aren’t delayed.

5. This delay allows “search cancel” messages to catch up, mitigating flooding.
Delayed Response to Untrusted Friends

1. Delayed response to queries from untrusted peers.

2. **Basic timing attack is prevented.**
   - Receiving a reply in less than 150 ms would reveal the responder as a data source to potentially untrusted peers.

3. Delay value chosen between **150-300 ms** to emulate the delay of a longer path.
Delayed Response to Untrusted Friends

1. Delayed response to queries from untrusted peers.

2. **Basic timing attack is prevented.**
   - Receiving a reply in less than 150 ms would reveal the responder as a data source to potentially untrusted peers.

3. Delay value chosen between **150-300 ms** to emulate the delay of a longer path.
Delayed Response to Untrusted Friends

1. Delayed response to queries from untrusted peers.

2. **Basic timing attack is prevented.**
   - Receiving a reply in less than 150 ms would reveal the responder as a data source to potentially untrusted peers.

3. Delay value chosen between **150-300 ms** to emulate the delay of a longer path.

- **T**, untrusted friend of **P** is source
  - Response Time = RTT + 270

- **S**, untrusted friend of **T** is source (2 hops from querier)
  - Response Time = RTT + q + 2*5 + r = RTT + 430

Diagram:
- **Query originator** connected to **T** with a 5 ms link.
- **T** connected to **S** with a 270 ms link.
- **S** connected to **P** with a 5 ms link.
- q = 150 ms.
OneSwarm: Anonymity Mechanisms

Delayed Response to Untrusted Friends

1. Delayed response to queries from untrusted peers.

2. **Basic timing attack is prevented.**
   - Receiving a reply in less than 150 ms would reveal the responder as a data source to potentially untrusted peers.

3. Delay value chosen between 150-300 ms to emulate the delay of a longer path.

- **T, untrusted friend of P** is source
  - Response Time = $\text{RTT} + 270$
- **S, trusted friend of T** is source and is 2 hops away from querier
  - Response Time = $\text{RTT} + q + 2 \times 5 + r = \text{RTT} + 160$

\[ q = 150 \text{ms} \]
## OneSwarm: Trust Relationships

<table>
<thead>
<tr>
<th>TRUSTED FRIENDS</th>
<th>UNTRUSTED FRIENDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>No delays when responding to queries from trusted friends.</td>
<td>Delays introduced for queries from untrusted friends.</td>
</tr>
<tr>
<td>Can see each others’ file lists.</td>
<td>Cannot see each other’s file list.</td>
</tr>
</tbody>
</table>
Outline

- Forensics applied to P2P filesharing systems ✔
- OneSwarm Overview ✔
- Attacks on OneSwarm
- Lessons learnt
Three Attacks that Allow Forensic Investigation

- We discovered three weaknesses in OneSwarm that allow forensic investigation of CP crimes:

1. We identify a new **timing attack**.

2. Correct **collusion attack** analysis to include file popularity.
   - For comparison: “achieving 95% precision requires that at least $k = 6$ attackers; chances of success, when $C = 30$ of the $N = 1000$ peers are attackers is much less than 1%.”

3. Show novel application of a **TCP-based attack**.
   (not in this talk; detailed in paper.)
Timing Attack: Problem Statement

**Scenario A**  OR  **Scenario B**  

<table>
<thead>
<tr>
<th>Scenario A</th>
<th>OR</th>
<th>Scenario B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search Origin</td>
<td>Data Source</td>
<td>Search Origin</td>
</tr>
<tr>
<td>A1 -----------</td>
<td>T -----------</td>
<td>A1 -----------</td>
</tr>
<tr>
<td>X ------------</td>
<td>r=300ms ------</td>
<td>X ------------</td>
</tr>
<tr>
<td>r=300ms ------</td>
<td>q=150ms ------</td>
<td>r=150ms ------</td>
</tr>
</tbody>
</table>

Data Source
Timing Attack: Problem Statement

**Scenario A**

- **T is source:**
  - Response Time, $\delta$
  - $\delta = \text{RTT}_X + r$
  - $(\delta - \text{RTT}_X) \leq 300\text{ms}$

**Scenario B**

- **S is source:**
  - S is untrusted friend of T
  - Response Time, $\delta$
  - $\delta = \text{RTT}_X + q + \text{RTT}_Y + r$
  - $(\delta - \text{RTT}_X) \geq 300\text{ms}$
Timing Attack

- Real system has undocumented steps and delays
- But the attack still holds.
- The graph shows the attack carried out on a LAN.
Timing Attack: Trusted Friend Scenario

- \( S \) is source:
- \( S \) is trusted friend of \( T \)
- Response Time, \( \delta \)
- \( \delta = \text{RTT}_X + q + \text{RTT}_Y + r \)
- \((\delta - \text{RTT}_X) = 150\text{ms} + \text{RTT}_Y\)
  - This quantity can be less than or greater than 300!
**Inference From Attack Result**

<table>
<thead>
<tr>
<th>Attack Result</th>
<th>Attacker’s Conclusion</th>
<th>Actual Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta - \text{RTT} \leq 300\text{ms}$</td>
<td>Target is the source</td>
<td>Either target or its trusted friend is the source</td>
</tr>
<tr>
<td>$\delta - \text{RTT} &gt; 300\text{ms}$</td>
<td>Target is not the source</td>
<td>Target is not source</td>
</tr>
</tbody>
</table>

- **Net result:**
  - Sources are always detected correctly. (No False Negatives)
  - Trusted friend of source is sometimes detected as source.
    - This is not a false positive in context of criminal liability.
Trusted Relationship Raises Criminal Liability

- OneSwarm peer *knowingly* acts as a proxy for its trusted friend’s shared files.
- OneSwarm peer can see the filenames (very explicit!) of its trusted peers in the GUI. Target is distributing CP with *knowing intent*.
  - *See 18 U.S.C. § 2252A(b)(1).*
- Setting a trusted relationship gives target a non-pecuniary benefit of performance, which incurs greater punishment.
- Evidence found at the target supports a search warrant of the trusted peer that is the actual source of CP.
Timing Attack: Feasibility and Success

- Requires just one attacker, all its neighbors are targets of investigation but one at a time
- Given network of $N$ peers of which $C$ are attackers:
  - **Hypergeometric CDF** gives probability that at least $k$ of the $U$ untrusted friends assigned by the Community Server are attackers.
- E.g., if attackers comprise 10% of OneSwarm network, 90% of the remaining peers are connected to one attacker.

- Attack has zero false positive rate and a 100% precision given that forensics goal is
  - to identify peers that either share CP files themselves
  - or are conspirators of the actual sources.
Does RTT variance affect attack success? No.

![Graph showing trials to achieve a false positive rate < 0.0001 as a function of additional RTT standard deviation (ms)].

- Mean RTT = 150ms
- 34 trials for RTT std.dev of 1500ms
Timing Attack Defenses

- We prove that the attack is defeated if delays enforced by OneSwarm are modified.
  - **Option 1:** increase source response delay, $r$?
    - Yes, but delays are 4 times worse than Onion Routing.
  - **Option 2:** decrease query forwarding delays, $q$?
    - Yes, but then search cancel messages don’t prevent flooding of queries to entire network.

- Detailed proof in paper; based on constraint satisfaction.

- In the end, to defeat the attack, minimal delays must be enforced that are higher than Onion Routing.
  - Reestablishes the tradeoff between privacy and performance.
  - OneSwarm does not achieve both.
Collusion Attack

- Queries are sent to all neighbors
- If neighbor has the content:
  - query is **not** forwarded
- If doesn’t have content:
  - Forwarding only with prob $p$.
- One attacker can’t determine if T has content or just didn’t forward.
- Smaller p: higher FPR, but harder to find content.
  - $p=0.5$ in the paper
  - $p=0.95$ in the source code.
Collusion Attack Classification Test

- **Test Setting**: $k$ colluding attackers get connected to a target. One of them queries the target.
- **Test**: Do no colluders get a forwarded query?

Twin Timing Attack was deterministic; this is probabilistic.
- Prior probability of file possession is important.
Precision and False Positive Rate

- Given that the test concludes that T is the source, what is the probability that T is indeed the source?

\[
\text{Precision} = \frac{v}{v + (1 - v)(1 - p)^{k-1}} \quad \text{(Applying Bayes’ Rule)}
\]

\(\kappa\): Number of colluding attackers attached to the target

\(v\): Probability that T has the file (content popularity)

\(p\): Forwarding Probability

Probability that the test wrongly infers a non-source as source

\[fpr = (1 - p)^{k-1}\]
Collusion Attack: Success and Feasibility

\[ \phi \text{ is precision} \]

Prob. of forwarding \( p = 0.95 \)
Collusion Attack: Success and Feasibility

- 4 colluding attackers are sufficient to acquire probable cause evidence for content that is sourced at only one in a hundred peers.

- False Positive Rate is less than 0.0025.
Collusion Attack: Success and Feasibility

\( \phi \) is precision

Prob. of forwarding
\( p=0.5 \)
Collusion Attack: Success and Feasibility

Fraction of Peers that are attackers

Prob. of Attack Success

$k \geq 1$
$k \geq 2$
$k \geq 4$
$k \geq 6$
$k \geq 8$
$k \geq 10$
OR
Collusion Attack: Success and Feasibility

- For $p=0.5$ (paper)
  - if investigators comprise 25% of the network
    - 50% of peers vulnerable (80% prec.; 1/100 content)
  - If investigators comprise 15% of the network
    - 5% of peers vulnerable (80% prec.; 1/100 content)

- For $p=0.95$ (software)
  - If investigations comprise 10% of the network:
    - 35% of peers vulnerable (95% prec.; 1/100 content)
  - If investigators comprise 25% of the network
    - 98% of peers vulnerable (95% prec.; 1/100 content)

- Investigators can easily rejoin.
Developer’s Response

- Informed developers in May 2011
- In August, new version of OneSwarm was released
- It does not enforce delays required to defeat Timing Attack
- It has a reduced the probability of forwarding queries, back to p=0.5
  - Mitigates collusion attack, but does not prevent it for popular content
- It was suggested to blacklist all Tor IPs, PlanetLab IPs, and UMass IPs. (http://forum.oneswarm.org/topic/1927)
Conclusion

- We show that plain view evidence sufficient for probable cause can be acquired from OneSwarm.
  - Following a constrained criminal procedure.
  - Sybil attack increases effectiveness of single investigator.
  - Leveraging “trusted friends” introduces criminal liability rather than protecting privacy.

- Our work proves that OneSwarm design does not suffice to achieve complete anonymity and privacy preservation.
  - There is a tradeoff between privacy and performance which OneSwarm does not overcome.

- We introduced a new threat model, based on digital forensics and computer crime law.
  - All anonymity systems should take legal procedure and definitions into account when protecting privacy.
Appendix-a: Developers’ Response

- http://forum.oneswarm.org/topic/1927

Piatek is working on a defense against the search flooding but he is busy with some other stuff right now so I’ll have to look at that and merge it into a future release.

In the meantime there are a couple things you can do to keep them away from your community server (with some risk for collateral damage):

1. In the firewall, block:
   * UMASS-NET (NET-128-119-0-0-1) 128.119.0.0 - 128.119.255.255
   * All planetlab nodes: http://oneswarm-support.appspot.com/plab.latest
   * All tor nodes: http://oneswarm-support.appspot.com/tor.latest

2. Limit key registration per ip to less than 5 (maybe 2?) in the community server conf.
   # The default number of keys that can be registered by a single IP.
   #key.registration.limit.ip.default=5
   ## The default key registration limit per account.
   #key.registration.limit.account.default=5

EDIT: I’m not sure where the attacking nodes are hosted, but these are all IPs they have so you could block all of these:
   http://www.fixedorbit.com/cgi-bin/cgirange.exe?ASN=1249

POSTED 1 MONTH AGO #
Appendix-b: Proof Outline  (Privacy-Performance Tradeoff)

- Constraint based system of (in)equations
  - Constraint 1: Basic timing attack must be impossible
    \[ \text{min}(r) = 1 \text{rtt} = 2l \quad \text{(avg value taken)} \]
  - Constraint 2: Twin Timing Attack must not work
    \[ \max(\text{SumA}-(\text{RTT1}+\text{RTT2})) > \min(\text{SumB}-(\text{RTT1}+\text{RTT2})) \]
    \[ \Rightarrow \ max(r) > \ min(q)+\min(r)+2l \]
  - Constraint 3: For a querier’s cancel messages to stop all instances of the query message within \( h \) hops of querier,
    \[ \max(r)+2l + h l < h(\min(q) + l) \]
    \[ \Rightarrow \ max(r)+2l < h(\min(q)) \]
    \[ \Rightarrow \ min(q) > (\max(r)+2l)/h \]
  - Constraint 4: For a graph where a peer’s outdegree is at least 39, \( h=2 \).
    \( h=3 \Rightarrow \) query reaches 59000 peers.
Appendix-c: comparison with O.R.

- Our analysis of **OneSwarm** with modified delays that thwart timing attack:
  - Expected time $t$ to receive a response from a source $x$ hops away is $E[t] = 8xl$.
    - For $x=1$, $E[t] = 8l$
    - For $x=3$, $E[t] = 24l$

- **Onion Routing**: with chain of 3 proxies, the delay in receiving data from a Torrent search engine is $E[t] = 6l$.

- Tradeoff between Privacy and Performance.
  - OneSwarm does not achieve both.