Defeating Vanish with Low-Cost Sybil Attacks Against Large DHTs

Scott Wolchok\textsuperscript{1} Owen S. Hofmann\textsuperscript{2} Nadia Heninger\textsuperscript{3} Edward W. Felten\textsuperscript{3} J. Alex Halderman\textsuperscript{1} Christopher J. Rossbach\textsuperscript{2} Brent Waters\textsuperscript{2} Emmett Witchel\textsuperscript{2}

\textsuperscript{1} The University of Michigan \textsuperscript{2} The University of Texas at Austin \textsuperscript{3} Princeton University
Road Map

1. What is Vanish?
2. Attacking Vanish
3. Costs and performance
4. Countermeasures
5. What went wrong?
Why Self-Destructing Data?

“Transient” messages tend to persist

Stored copies enable **retroactive attacks**

Attacker subpoenas data months or years later
Vanish and Vuze

Vanish uses the Vuze DHT (Distributed Hash Table)
Over 1 million nodes, mostly BitTorrent
Nodes delete values after 8 hours
Vanish and Vuze

Shares placed at random locations in the DHT
Replicated to 20 “closest” nodes
Is Vanish Secure?

Vanish 0.1 prototype released at publication

Included user-friendly Firefox plugin

Focused wide attention on its practical security
1. What is Vanish?

2. Attacking Vanish

3. Costs and performance

4. Countermeasures

5. What went wrong?
DHT Crawling Threat

**Threat:** attacker might continuously archive *all* data in the DHT

Later, query archive to decrypt messages

Don’t need specific targets when recording
Crawling with a Sybil Attack
Making the Attack Practical

Insight: have **8 hours** to observe fragments

Vuze replicates to 20 nearest nodes

1. Every 30 minutes
2. On join!
“Hopping” Strategy

Sybils “hop” to new IDs every 3 minutes

160x resource amplification over 8 hours

Practical attack needs only ~2000 concurrent Sybils with hopping
Making the Attack Practical

Insight: Vuze client is a notorious resource hog

Only 50 instances fit in 2 GB of RAM!

Can we more efficiently support 2000 Sybils?
Optimized Sybil Client

C, lightweight, event-based implementation

Listen-only (no Vuze routing table!)

Thousands of Sybils in one process
Road Map

1. What is Vanish?
2. Attacking Vanish
3. Costs and performance
4. Countermeasures
5. What went wrong?
Attack Costs?

Vanish paper estimate (for 25% recovery at $k=45$, $n=50$):

– 87,000 Sybils
– $860,000/\text{year}$

What does attacking Vanish really cost?
Experiments

1. Insert key shares into the DHT
2. Run attack from 10 Amazon EC2 instances
3. Measure:
   - DHT coverage = % key shares recovered
   - Key coverage = % messages decrypted
   - Attack cost = EC2 charges (Sep. 2009)
## Experimental Results

### Cost for >99% Vanish key recovery?

<table>
<thead>
<tr>
<th>Attack</th>
<th>Concurrent Sybils</th>
<th>Key Shares Recovered</th>
<th>Annual Attack Cost*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hopping</td>
<td>500</td>
<td>92%</td>
<td>$23,500</td>
</tr>
<tr>
<td>Hopping + Optimized Client</td>
<td>2000</td>
<td>99.5%</td>
<td>$9,000</td>
</tr>
</tbody>
</table>
DHT Coverage vs. Attack Size

- DHT Coverage
- Effective Sybils

- Experiment
- Model

Hopping plus Optimized Client
Key Recovery vs. Attack Size

Key-sharing parameters ($k/n$)

- Blue line: 7/10
- Red line: 9/10
- Green line: 45/50

Key Recovery:
- 25% @ 70k Sybils
- 99% @ 136k Sybils

Effective Sybils: 0, 30k, 60k, 90k, 120k, 150k, 180k, 210k, 240k, 270k

Hopping plus Optimized Client
Annual Cost vs. Key Recovery

- 25% @ $5000
- 90% @ $7000
- 99% @ $9000

Key-sharing parameters $(k/n)$
- 7/10
- 9/10
- 45/50

Hopping plus Optimized Client
Storage

$1400/yr for all observed data

$80/yr for potential key shares
Road Map

1. What is Vanish?
2. Attacking Vanish
3. Costs and performance
4. Countermeasures
5. What went wrong?
Increase Key Recovery Threshold?

Required coverage increases in $n$ and $k/n$

Why not raise them? (99/100?)

**Reliability**: some shares lost due to churn

**Performance**: pushing shares is slow!
Limit Replication?

Attack exploits aggressive replication

Less replication might make the attack harder, but how much?

More in a few slides...
Sybil Defenses from the Literature?

Client puzzles

Limit ports/IP, IPs/subnet, etc.

Social networking
Detecting Attackers

Find and target IPs with too many clients

Use node enumerator, *Peruze*

Can detect attack IPs hours after the attack

Detected the Vanish demo
Road Map

1. What is Vanish?
2. Attacking Vanish
3. Costs and performance
4. Countermeasures
5. What went wrong?
Cost Estimation Issues

Vanish paper extrapolated from 8000-node DHT

Assumed Sybils must run continuously

Assumed attacker uses inefficient Vuze client
Cost Not Linear in Recovery

![Graph showing key recovery fraction vs coverage fraction for different key-sharing parameters (k/n). The graph indicates that the cost of recovery is not linear, with different curves for 7/10, 9/10, and 45/50.](attachment:image.png)
Response to Our Work

Second report and prototype by Vanish team

New defenses

– Use both Vuze DHT and OpenDHT
– Disable replicate-on-join in Vuze
– Use less aggressive “threshold replication”

Will these defenses stop real attackers?

Conclusion

Showed attacks that defeat Vanish 0.1 in practice for $9000/year
Vanish team has proposed new defenses
Future work: are new defenses effective?

Our take: building Vanish with DHTs seems risky.
Defeating Vanish with Low-Cost Sybil Attacks Against Large DHTs

Scott Wolchok\textsuperscript{1}  \textbf{Owen S. Hofmann}\textsuperscript{2}  Nadia Heninger\textsuperscript{3}  Edward W. Felten\textsuperscript{3}  J. Alex Halderman\textsuperscript{1}  Christopher J. Rossbach\textsuperscript{2}  Brent Waters\textsuperscript{2}  Emmett Witchel\textsuperscript{2}

\textsuperscript{1} The University of Michigan  \textsuperscript{2} The University of Texas at Austin  \textsuperscript{3} Princeton University

http://z.cs.utexas.edu/users/osa/unvanish/
Vanish Attack Model

Need to recover $k$ of $n$ fragments

$p = \Pr\{\text{recover key fragment}\}$

$\Pr\{\text{recover VDO}\} = \Pr\{\text{recover } k \text{ or more fragments}\}$

Binomial distribution

$\Pr\{\text{recover VDO}\} = \sum_{i=k}^{n} \binom{n}{i} p^i (1 - p)^{n-i}$
Coverage Model

$m$ Sybils see $c$ of $N$ objects

Balls-in-bins problem

Expected fraction $= 1 - e^{-cm/N} = 1 - e^{-sm}$

$s = c/N$ is the (overlapping) fraction of the network observed by each Sybil
Prior Work

• Enumerating DHT nodes
  – Cruiser [Stutzbach 2006a,b]
  – Blizzard [Steiner 2007a]

• Measuring DHT traffic
  – Mistral [Steiner 2007b]
  – Montra [Memon 2009]
## Hopping plus Optimized Client

<table>
<thead>
<tr>
<th>Concurrent Sybils</th>
<th>Hours</th>
<th># VDO Fragments</th>
<th>Fragments Found</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>8</td>
<td>1650</td>
<td>1640 (99.4%)</td>
</tr>
<tr>
<td>2000</td>
<td>7.5</td>
<td>1700</td>
<td>1692 (99.5%)</td>
</tr>
<tr>
<td>500</td>
<td>8</td>
<td>1650</td>
<td>1561 (91.8%)</td>
</tr>
</tbody>
</table>