Intrusion Detection and Response in LOCKSS

Rachel Greenstadt
greenie@eecs.harvard.edu

Harvard University
Advisors: HT Kung and Mike Smith
September 30, 2004
Overview

- Problem: Electronic Archiving
- Approach: LOCKSS project
- Security Threats and Countermeasures
- Intrusion Detection and Response: Alarms
  - Do we need alarms?
  - Can alarms catch intrusions?
  - Can we take local steps to correct the network after an intrusion?
  - Can we catch intrusions faster?
A Crisis in Archiving

- Libraries know how to cooperate to preserve paper data
- With move to electronic data, publishers offer subscriptions
  - Offer “perpetual access,” but no business model for this
- Libraries want to preserve their own data BUT
  - Digital preservation is hard
  - (and most people don’t believe it)
Why is Digital Preservation Hard?

- Storage media are unreliable in the long term
  - MTBF of components
- Human error
- Many anecdotes of backup failures
  - Suggest bit rot/failures happen in Byzantine ways
  - Almost everyone has a story
  - Companies don’t like to talk about it
Overview

- Problem: Electronic Archiving
- Approach: LOCKSS project
- Security Threats and Countermeasures
- Intrusion Detection and Response: Alarms
  - Do we need alarms?
  - Can alarms catch intrusions?
  - Can we take local steps to correct the network after an intrusion?
  - Can we catch intrusions faster?
The LOCKSS Approach

- Build P2P community of libraries
- Each library maintains a replica of the Archival Unit (AU)
- Goal: Maintain consensus on content of AUs
- Libraries help each other repair replicas
  - Audit and repair detected damage with voting
- Publishers provide content for archiving
  - Without responsibility for preservation
Why LOCKSS?

- LOCKSS is a real project
  - Deployed version
  - Consortia of libraries
  - Engineering team
- Good vehicle to study intrusion detection for other P2P/distributed systems
Peer Relationships

LOCKSS has an internal PKI. If I’m a LOCKSS peer, other peers are:

- Friends—Peers with which I have out-of-band trust relationships. Friends sign certificates for each other.
- Discovered Peers—These peers form my trust web.
- Reference List—A subset of discovered peers that can be polled.
- Undiscovered Peers—Peers unknown (or untrusted) by me.
Opinion Polls

Periodically, peers poll a subset of the reference list and compare the votes to their local AU

- Landslide agreement
  - Do nothing.

- Landslide disagreement
  - Repair

- Inconclusive
  - Alarm!
Updating the Reference List

- Before voting, peers nominate other peers for inclusion in reference list/trust web.
- At the end of poll, voting peers are purged from reference list
- Add some friends
- Add some nominees
Overview

- Problem: Electronic Archiving
- Approach: LOCKSS project
- Security Threats and Countermeasures
- Intrusion Detection and Response: Alarms
  - Do we need alarms?
  - Can alarms catch intrusions?
  - Can we take local steps to correct the network after an intrusion?
  - Can we catch intrusions faster?
Security Concerns

Basically giving away write permission on your archive!!!!

- For LOCKSS to be useful, benefits must outweigh costs
Alarms

- Idea
  - Simultaneous bit rot is rare
  - If voters disagree, adversary activity likely
- LOCKSS didn’t specify response
  - This is my contribution
My Contributions

- Ran experiments to verify alarms were needed and could detect intrusions
- Devised a localized protocol to respond to alarms by *Healing* compromised peers
- Ran simulations to evaluate and tune this protocol
- Devised and tested an augmented protocol to trigger alarms earlier in an attack
Evaluation Measures

- Iterative process of simulation and reasoning about the system design and simulation results
- Proofs would be nice, but system complexity would render them inaccurate or intractable
  - Problem with many P2P systems
- Initially, goal was to keep adversary from damaging > 50% of the AUs, reached that, see how close we can get to 0.
Simulations

- Simulate 1000 peers participating in the LOCKSS system
- Each peer has one AU that can be good or bad.
- Some variable fraction of these peers are adversarial.
- Adversary follows the strategy of lurk and try to get a presence on good peers’ reference lists, then attack when they’ll win decisively.
## Static Variables (Assumptions)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Peers</td>
<td>1000</td>
</tr>
<tr>
<td>Sybils</td>
<td>200</td>
</tr>
<tr>
<td>Topology</td>
<td>Cluster</td>
</tr>
<tr>
<td>Poll Size</td>
<td>10-20</td>
</tr>
<tr>
<td>Supermajority</td>
<td>70%</td>
</tr>
<tr>
<td>Reference List Goal</td>
<td>60</td>
</tr>
<tr>
<td>Churn Ratio</td>
<td>10%</td>
</tr>
<tr>
<td>Lurktime</td>
<td>3600 of 7200 ticks</td>
</tr>
<tr>
<td>MTBF Doc</td>
<td>$\frac{1}{200}$ yrs</td>
</tr>
</tbody>
</table>
Dynamic Variables

- # of Adversarial Peers (10-400)
- Random seeds
- Alarm Response
Overview

- Problem: Electronic Archiving
- Approach: LOCKSS project
- Security Threats and Countermeasures
- Intrusion Detection and Response: Alarms
  - Do we need alarms?
  - Can alarms catch intrusions?
  - Can we take local steps to correct the network after an intrusion?
  - Can we catch intrusions faster?
Do we need alarms?

Figure 1: MUTE alarm: when an alarm is called, do nothing (proceed as if we won the poll)
Can alarms detect intrusions?

Theory—Why do alarms occur?

- Alarm might happen if an adversary tries to win a poll without enough votes (can’t count on this)
- Peers with corrupted copies + peers with good copies => alarms
- Adversary can’t corrupt enough copies at once to win without giving alarms a chance to fix things
  - Arrange reference list updating and rate limiting to make this so
Can alarms detect intrusions?

Figure 2: Foothold ratio for a very patient adversary and results after the first alarm.
Alarms

- Do we need alarms?
- Can alarms catch intrusions?
- **Can we take local steps to correct the network after an intrusion?**
- Can we catch intrusions faster?
How can we respond?

- Change our state (not enough)
- Ask our friends to change their state
- Ask our friends to ask their friends · · ·
## Peer states

<table>
<thead>
<tr>
<th></th>
<th>Admin</th>
<th>Computer</th>
<th>AU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Healthy</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Damaged</td>
<td>Good</td>
<td>Good</td>
<td>Bad</td>
</tr>
<tr>
<td>Subverted</td>
<td>Good</td>
<td>Bad</td>
<td>Good/Bad</td>
</tr>
<tr>
<td>Evil</td>
<td>Bad</td>
<td>Bad</td>
<td>Good/Bad</td>
</tr>
</tbody>
</table>

**Figure 3: Classification of Peers**

- We can heal subverted peers and revoke certificates of nominated evil peers
Healing Alarm Procedure

- Contact friends in trust web, ask them to check for compromise and patch. Treat all unhealed peers as undiscovered.
- What if we want to do more than just our friends?
  - We can heal nodes at depth 2, by asking friends to ask them to heal themselves
  
Healed peers revoke certificates they signed when subverted
Types of Alarms

(a)

(b)

(c)
Healing Results 1

Figure 4: Doing nothing, compared to healing d=1 (end of simulation)
Healing Results 2

Figure 5: Doing nothing, compared to healing d=1 (worst pt of simulation)
Healing Results 3

Figure 6: Doing nothing, compared to healing $d=1$
Results with Evil Peers

Figure 7: Experiments with 600 loyal peers and 400 malign peers, run with 4 seeds and varying numbers of evil peers.

Effect of Trusted Evil

- Green: Damaged
- Blue: Subverted
- Red: Evil

Number of Bad Copies

Number of Trusted Evil Peers (of 400 total)
Imperfect Healing \( p = 0.5 \)

**Figure 8:** \( d=1, p=0.5 \), compared to healing \( d=1, p=1 \)

What if the sys admins are unable to heal all subverted nodes?
Healing With Depth 2

Figure 9: \(d=2, p=1\), compared to healing \(d=1, p=1\)
Depth 2, Prob 0.5

Figure 10: $d=2, p=0.5$, compared to healing $d=1, p=0.5$
Benefits of Localized Alarms

- The system does not have to halt for human intervention for every alarm.
- A Nuisance Adversary might try to DOS the system by calling many alarms.
  - In the local alarm case these adversaries’ victims will be their friends.
  - These friends can revoke their certs if they are too obnoxious, so self-policing.
Alarms

- Do we need alarms?
- Can alarms catch intrusions?
- Can we take local steps to correct the network after an intrusion?
- **Can we catch intrusions faster?**
Adding Hashing to LOCKSS

- Old idea to use local hashing to augment LOCKSS
- Try a simple idea to see how much it can help
  - Each peer keeps a local hash of its AU
  - If poll result is repair, call alarm if hash matches the document
- Current research on more sophisticated use of hashing
Hashing Results

Figure 11: Depth 1, prob 1, keeping a hash.
More Hashing Thoughts

- What happens when the hash goes bad?
- If you have a good doc and a bad hash (or a bad document) then you are still vulnerable.
- Current research to mitigate this issue...
Conclusions

- Alarms are necessary and can be effective
- Local healing alarms can recover from the compromise of up to 40% of LOCKSS peers
- Adding hashing techniques to LOCKSS is effective
- A combination of hashing and alarms can mitigate the risks involved in LOCKSS
Future Directions

- Hashing
- Periodic re-infections
- Adversaries that go back into lurk mode
Reset Results

Figure 12: Compare resetting the ref list to doing nothing
Some Other Approaches

- Other approaches can be complementary
- RAID/backups
- Use hashing (handwave)
- Have peers store signed hashes for each other
  - Reasonable, but won’t achieve consensus
  - Incentive issues
  - LOCKSS models current library interactions
- But LOCKSS is happening!
- End of talk: look at ways to add redundancy to LOCKSS with hashing