An Architecture for Developing Behavioral History

Mark Allman (ICSI)
Ethan Blanton (Purdue), Vern Paxson (ICSI)

Network Chat, CWRU, April 2005

"Ligthen up while you still can,
Don't even try to understand"

Background

- Goal: we want to prevent unwanted traffic on the Internet
  - DDoS, worms, spam, scanners, etc.
  - *think in terms of commonality*

- Problem: interactions on the network are largely anonymous and self-contained
  - Sort of true

Background (cont.)

- Say some host *S* wants to access some network service on your host or network.

- What can we use to decide whether to service this request from this requester?

Background (cont.)

- Information about remote systems:
  - IPsec or MD5
  - Local cache of previous activity (e.g., developed by a site’s IDS)
    - Remote information from a friend (trusted IDS interaction)
  - Centralized databases about previous activity (e.g., www.dshield.org)
  - List of host "type" in some central list (www.sorbs.net)

Background (cont.)

- All the information is either:
  - *narrow* in scope
  - *difficult* to obtain/setup

Goal

- Devise a system for accumulating reports about unwanted traffic:
  - Internet-scale
  - handles arbitrary "unwanted" traffic
  - distributed
  - policy independent
  - open

- Proposed as a community project because its big and complicated and could use help from smart people.

Goal (cont.)

- Is this an important or worthy goal?

- Example: Slammer
  - Single packet UDP worm
  - Infected 75,000 hosts (in 10 minutes!)
  - Peak aggregate scanning rate: 55M scans / second

Goal (cont.)

- Each infected Slammer box scanned at 733 scans / second, on average
- To hit a single address in every /24 in the IPv4 address space would take 6 hours
- On the other hand, every second 733 machines were potentially able to figure out that some given machine was infected
- (Don’t think too deeply about this example, there are holes.)

Goal (cont.)

- The worm was fast ... but, the space is large.

- The key point is that *someone* knows about a bad actor *quickly* and could potentially save subsequent infection by alerting others.

The Plan

- A distributed database:
  - reports are inserted into the database when unwanted activity is detected
  - providers of services can look up information about previous activity when determining whether to service a request

- Build on a DHT:
  - e.g., OpenHash
  - robust, distributed

The Plan (cont.)

- Insert records with two hash keys:
  - `insert (bad_actor, report)`
  - `insert (my_public_key, report)`

- Keys ... the kiss of death!
  - Not so... in our system they are used only to correlate reports from the same entity
  - Not tied to identity
  - No "PKI"

The Plan (cont.)

- Report types:
  - behavior reports
  - witness statements
  - signatories

Behavior Reports

- We focus on attacks; could focus on other aspects of behavior

- Insert record with:
  - timestamp
  - actor identity
  - protocol and port number (optional)
  - behavior observed
  - behavior digest
  - signature

Behavior Reports (cont.)

- When are these records inserted?
  - reporter’s discretion
- How much "evidence" is required to insert something into the database?
  - reporter’s discretion
- Who polices the entries?
  - nobody
  - and everybody
- How can this possibly work?
  - maybe it cannot
  - will need a bunch of work

Behavior Reports (cont.)

- On face value behavior reports look dangerous and could be used in an attack themselves
  - e.g., someone's enemy reports some fictitious behavior in the hopes that firewalls will pick up on the behavior and block traffic

- So, we need to do some more work ...

Witness Statements

- It'd be nice if we could generate an *audit trail* that offered evidence that some report was not completely cooked up

- E.g., if routers along the path kept packet digests then they could be asked to enter a witness record into the database that reports that a given packet (part of the attack) was observed.

- A witness statement is not a judgment, but rather a statement of fact

Signatories

- Hosts that use particular records in making policy decisions can sign those records to indicate their use

- Much like the idea behind PGP
  - we build a web of trust
  - not quite the same because there is not hard and fast notion of *identity*

A Final Database Point

- What the database is not:
  - a determiner of *maliciousness*
  - a determiner of *policy*

- The database simply holds information and the determinations of the users

Policy

- When do we query the database for information?
  - querier’s discretion
- What do we do with all this information we have stuck into this DHT?
  - querier’s discretion
- Who’s reports can we believe?
  - querier’s discretion

Policy (cont.)

- The database provides a source of information that may or may not be used as part of *local policy decisions*
  - could deny access
  - could rate-limit access
  - could watch the traffic more closely
  - etc.

Trust

- The key problem with setting policy based on information from others is *trust*

- The information from the database may be wrong:
  - the reporter may have made an inaccurate assessment
  - the reporter may have intentionally lied
  - the information may be out-of-date

Trust (cont.)

- We address the problem of trust by using *locally-determined reputations*
- We can access aboth a reporter’s history and the "bad" actor’s history

Trust (cont.)

- We can assess the *reputation* of various reporters:
  - Do lots of entities *corroborate* some assessment?
  - Have many entities signed reports?
  - Does the audit-trail support the reported behavior?
  - Do we have local evidence that is consistent with the reported behavior?
  - (We might even know the identity of a reporter!)

- All these can be gamed, so we need research into reputation calculations.

Trust (cont.)

- Some work has been done on reputations in peer-to-peer systems.
  - E.g., Aberer and Despotovic scheme
    - can catch big-time cheaters
    - cannot catch low-rate cheaters

- To make a system like we're talking about work more work needs to be done in reputation assessment.

Cheating

- The whole problem is that we have to assume there will be bogus information in the database.

- Consider a motivated attacker:
  - Finds a well known and well trusted key
  - Simply mimics the activities of that key
  - This will lead to a solid reputation, whereby the attacker then possibly insert bogus reports that will be believed

  - Could be dealt with using the order of reports, the set of nodes reporting some behavior, witness reports

Cheating (cont.)

- An attacker could forge witness reports
- Maybe witnesses use well-known (e.g., ISPs) keys, so that not everyone can be a witness
  - An "expert witness" notion

Cheating (cont.)

- An attacker could fake signatories
- So, take into account the reputation of the signatories, as well

Cheating (cont.)

- An attacker could just flood the database with random records to increase the computational complexity of ferreting out the truth.

Issues

- There are a *ton* of issues ....

Deployment

- The system is incrementally deployable
  - doesn’t require global coverage or participation

Linking Keys and Identity

- There are pros and cons to associating keys and identity

- We don’t want to tackle the PKI problem
- We don’t want to require pair-wise key exchange
  - so, we need some notion of reputation

Linking Keys and Identity (cont.)

- What if keys are matched to identity by an adversary?
  - attacker can avoid detection by avoiding key's network
    - an incentive for deployment!
    - could work against containing a global outbreak
  - attacker could determine site's security policy
  - could be a means to embarrass an organization (e.g., because their assessments are often wrong)

Revoking Reports

- Is it worth being able to reverse a decision?
  - SSH example
- More speculatively... after an infected machine has been cleaned up.
- A must is to be able to note that a key has been compromised.

Openness

- We may need to abandon the completely open framework
  - DHT membership could be limited
    - think of this as a piece of infrastructure, e.g., a DNS server
  - reporters could be "somewhat known"
  - witnesses could be a "small" number of ISPs

- The entire system could be instantiated multiple times to be used in "closed" environments.

Overhead

- Even small networks like ICSI's are visited by thousands of hosts every day

- Lookup for every transaction?
  - computational burden
  - bandwidth burden
  - causes delay

- Cache?
  - well... maybe...

Surrogates

- Given the size of the job of gathering information and calculating reputations there seems to be room for help.
  - hosts on the network that constantly monitor the database, calculate reputations, etc.
  - make the information quickly and easily available via a web page
- However, this takes away *local* control, which is a fundamental notion to the system
  - the site could publish algorithms
  - the database is still available to everyone and so a site could periodically *audit* these surrogates

Conclusions

- Um....

Conclusions and Future Work

- We have sketched an architecture that we think the community could think about and implement (in some form)

- However, the entire talk has been future work