Improving Internet Security Through Cross-Organizational Information Sharing

Mark Allman, ICSI

Case EECS Seminar
October 2006

*"We like a long neck and a good old song.
Turn it up and then we’ll sing along."*

Collaborators

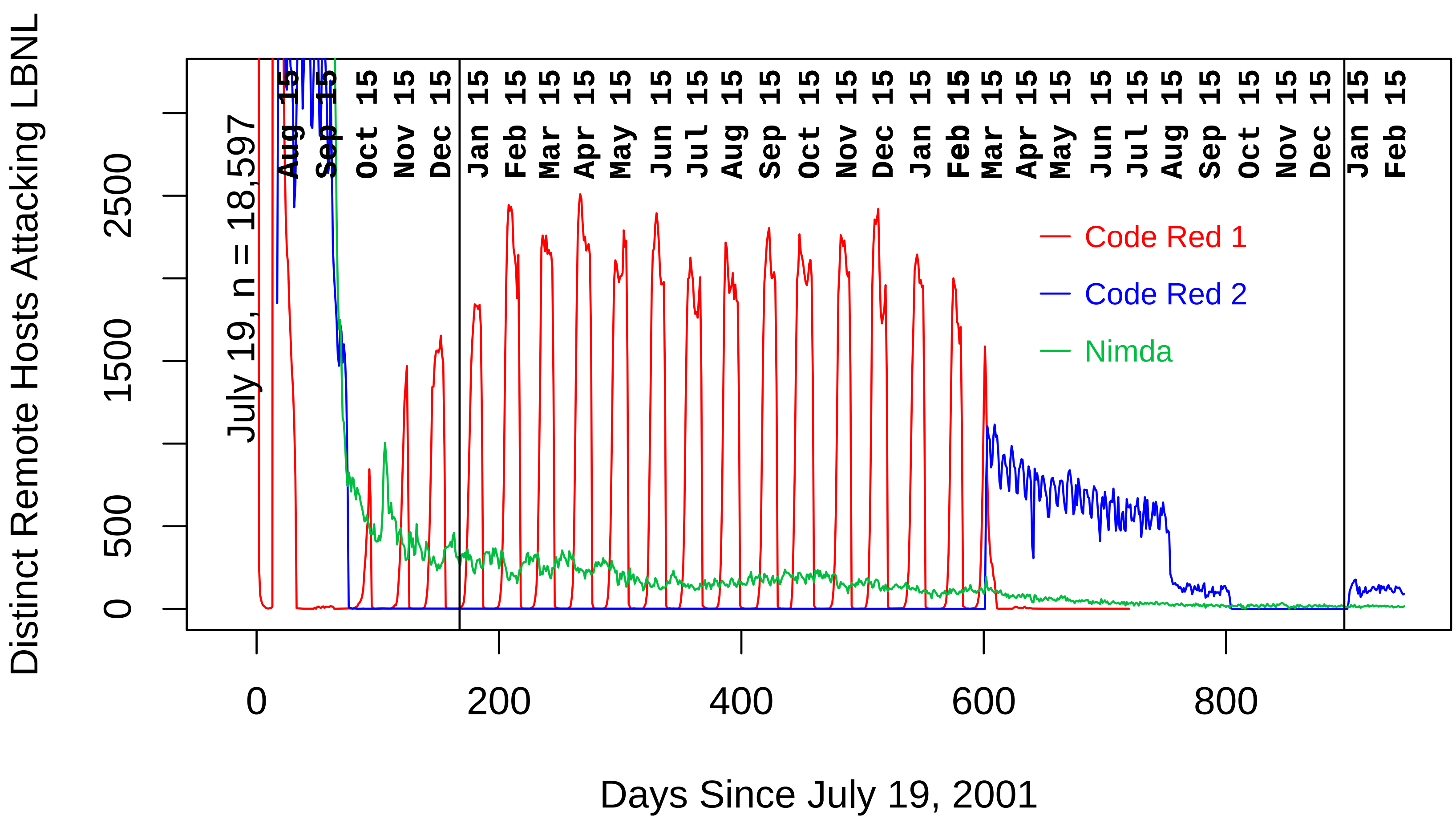
- Ethan Blanton, Purdue
- Vern Paxson, ICSI/LBNL
- Scott Shenker, ICSI/UCB

Background

- Network security is a mess
  - viruses, worms, spam, phishing, botnets, etc., etc.

- Always a new (and clever) attack
- Also, a multitude of *persistent* attackers
  - constant "background radiation"

Background (cont.)



(plot from Vern Paxson)

Background (cont.)

- Standard approaches:
  - firewalls
  - intrusion detection systems
  - virtual private networks
  - NATs
  - virus scanners
  - spyware cleansers

- Mostly single point mitigations
  - *whack-a-mole*

Basic Wonderings

- Can we improve the state of network security by sharing more information across organizations?

- Informal sharing exists -- especially in times of crisis
  - how much can we automate?

- How can we conduct information sharing in a routine and *practical* way that respects the possible sensitivity of the data?
  - e.g., due to user’s privacy concerns
  - e.g., due to a provider’s competitive concerns

Approach

- An open global system for sharing behavioral information about bad actors in the network
  - many vantage points sharing their view of the network
  - grass roots

- A problem with simply sharing observations is that the *linkages* between attacks is not taken into account
  - we propose a two-tiered system

A Question

- When some host sends a request to your HTTP server how do you know it is legit?

Analogies

- In other domains we assess service requester’s *track record*

- Mortgages
- Insurance
- Employment history
- Etc.

- Why not in networks?

A Database of Behavior

- There is no readily apparent history of a user or host’s activity on the Internet
  - interactions on the Internet are for the most part atomic

- This is a good thing
- This is a bad thing
- This isn’t even true

- We can all agree this is a *sensitive* area

A Database of Behavior (cont.)

- The problem with this is that each organization must learn of maliciousness by observing it
  - e.g., with an IDS
  - e.g., with a virus scanner
- So, let’s think about a *global* and *open* database that details the actions of *bad actors*
  - explicitly we exclude information about benign communication

Database Specifics

- Build a database using a global DHT
  - robust easy-to-use data structure

- E.g., with a simply get/put interface, ala OpenDHT

Database Specifics (cont.)

- Insertion:
  - IDS A:

    ```
    put (actor1,"actor1 is a scanner")
    ```

  - IDS B:

    ```
    put (actor1,"actor1 is a worm")
    ```

Database Specifics (cont.)

- Accessing:

```
history = get (actor1)

history ==>
  actor1 is a scanner
  actor1 is a worm
```

Database Specifics (cont.)

- Anyone can insert reports
  - whenever they figure out something about an actor
- Anyone can get a copy of the reports inserted into the database
  - input to policy
- Reports age out of the system in a reasonable amount of time
  - we have imperfect actor IDs like IP addresses
  - so, actors can "clean up"

Database Specifics (cont.)

- Immediate problem: how do we trust the information in the database?
  - bad actors could in fact put information into the database

Database Specifics (cont.)

- First, make the reporters identify themselves:

```
D = sign (key,"actor1 is a scanner")
put (actor1,D)
put (key_id,D)
```

- Now we can correlate reports from a given submitter

- No global PKI
- We assess a reporter's *reputation*

Reputations

- How do we assess reputations?
  - how many others say the same thing?
  - how does a given reporter's submissions compare with local observation?
  - are there witnesses that can validate that a given bit of traffic actually happened?
  - signatories: if someone finds a record useful in deriving policy they note it in the database

Cheating

- None of the above ideas provide ultimate trust
  - all probabilistic

- An attack on the system:
  - someone wants to insert bogus records
  - they observe a reporter with high reputation ... and simply report the same information under a new key
  - they have stolen a good reputation and have a vector to slip in small amounts of bogus information

- How do we catch this?

Cheating (cont.)

- Assess overlap of reports

- Assess timing
  - does one key always report first?
  - how long has a particular key been used? hours? weeks? months?

- *Ringers*

Cheating (cont.)

- Ultimately assessing the reputation of the reporter is a *local decision* for the information consumer

- A *crucial* area for research

Reputations II

- Bootstrapping scheme: web-of-trust

- Possibly workable in a global context, but a more open system would be preferable
  - e.g., reports from dark address hits to home users that wouldn’t be easily connected to a web-of-trust

Deployment

- Can be incrementally deployed
  - just providing *more information*

- Possibly heavy computational cost
  - can delegate work to surrogates
  - behavioral reports as a *service*

A Behavioral Database

- Lots of pieces are "around" and could be combined

- Need a way to assess reputations that provide reasonable assurance that large scale cheating is not happening
  - alternatively, abandon the notion that the database is open

- Devil is in the implementation details

A Behavioral Database (cont.)

- But, ...

- As sketched a behavioral database *expands* our view, but does not fundamentally change it

- In particular, we are left with little understanding into *coordinated* attacks

Coordinated Attacks

- An attacker compromises large groups of machines (bots)
- The hosts are loosely connected together to form a *botnet*
- Allows the attacker to leverage hundreds or thousands of machines in a single attack
  - makes attacks more severe
  - makes attacks more illusive

- The challenge then becomes to understand that a group of hosts is acting in a coordinated fashion

Contact Graphs

- Sekar, et.al. (Hotnets 2004) show how a who-talks-to-who contact graph (or, partial contact graph) can be used to find botnets

- Our goal is to make this idea more practical
  - focus on mis-behavior

The Long Arm of the Law

- Criminal activity in the real world is handled by expert law enforcement personnel

- Often assisted by untrained passers by who happen to have seen criminal activity by happenstance

- In the network we can have roughly the same state of affairs:
  - *detectives:* honeyfarms, sophisticated IDS systems, etc.
  - *witnesses:* NetFlow logs, web cache logs, etc.

Detectives

- Deep understanding of network
- Deep understanding of malicious behavior
- Small in number

- E.g., honeypot
  - understands how to get itself "infected"
  - can solidly assess the attacker’s actions

Detectives (cont.)

- Task: develop a *pattern* of malicious activity
  - attacked by host A
  - compromise via local SQL server (TCP port B)
  - outgoing fetch of malcode from host C via HTTP (port D)

- Distilled:
  - A:* -> X:B
  - X:* -> C:D
- Any host X that follows this pattern is likely to be similarly compromised

Witnesses

- No understanding of maliciousness or network security problems
  - simple observations of network activity
- Scattered widely throughout the network already
- Large in number

- E.g., NetFlow logs
  - accounting, provisioning, etc.

Witnesses (cont.)

- Task: dig through the activity log for patterns of activity provided by detectives

System Overview

- Use a small number of detectives to drive a large number of witnesses
- Detectives find a pattern of malicious behavior, query the witnesses for additional actors and formulate a picture of the botnet
- Witnesses simply provide information about the patterns that have been observed
  - no judgments made by the witnesses

System Overview (cont.)

- Detectives provide *depth*
- Witnesses provide *breadth*

System Overview (cont.)

- How do we trust the detectives?

- Detectives can be well known
  - important, because this system is easy to leverage for *fishing*

System Overview (cont.)

- How do we trust the witnesses?

- We don’t
  - too many witnesses to be well known
  - we want *ambiguity* in detectives queries
    - query only makes sense if witness has observed given pattern
  - witnesses cannot easily forge reports about non-existent traffic

Loose Private Matching

- Detective forms a hash with the given inputs
  - IP addresses, ports, protocol numbers, etc.
  - hash is not unique to one set of inputs

- Witness looks through logs for matching traffic

Loose Private Matching (cont.)

- Witness returns the hosts that match the pattern to the detective
  - returns all matches
    - witness cannot disambiguate collisions
  - result encrypted
    - use raw information as shared secret
    - if the information returned is due to a collision the data will be meaningless to the detective

Example

- Query: list the set of source IPs that have a given hash

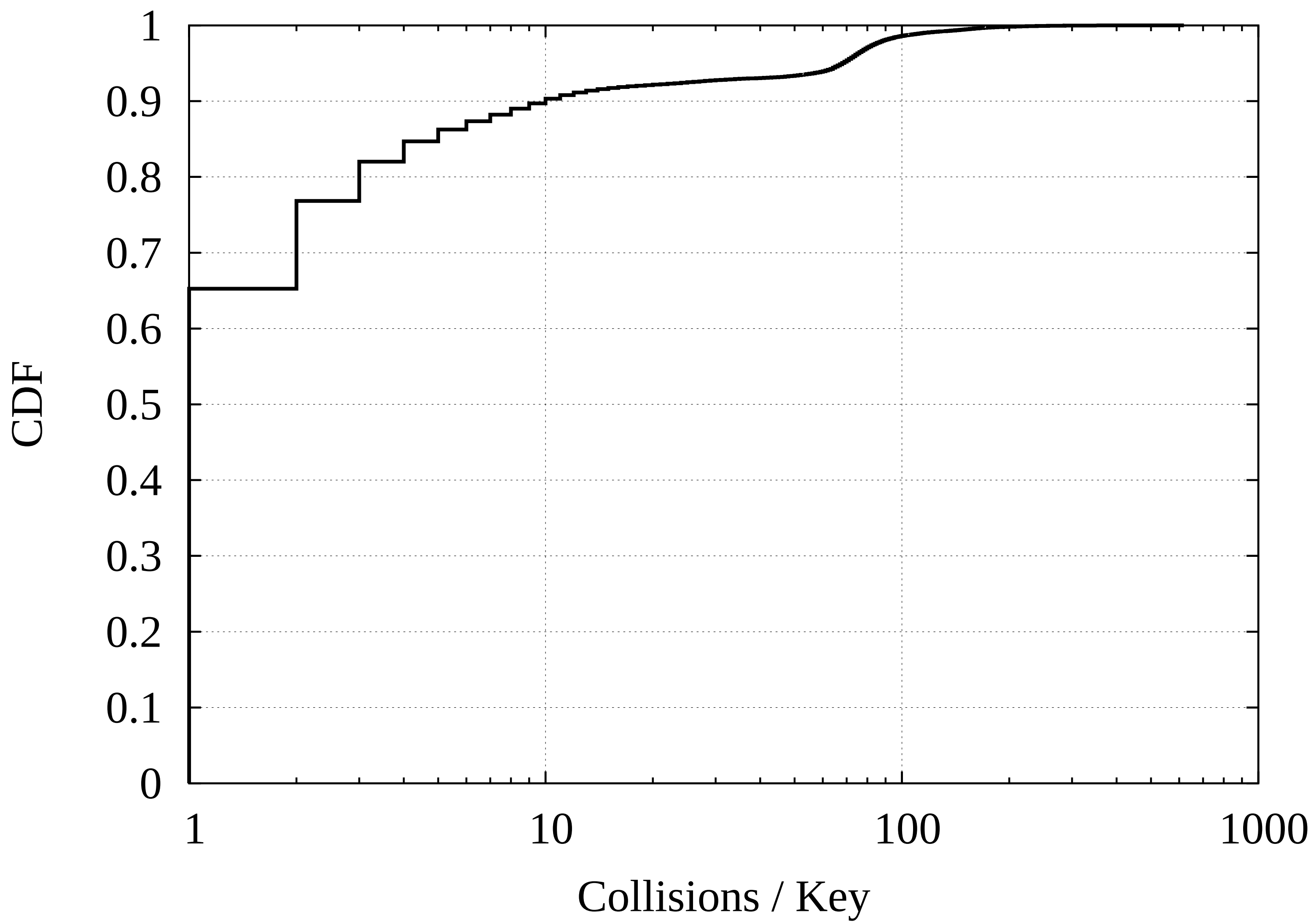
- Hash: byte-wise multiplication of:
  - destination IP address (4 bytes)
  - destination port number (2 bytes)
  - transport protocol number (1 byte)
  - (if a byte is zero, round to one)

- Given this simple formula, IP addresses *a.b.c.d* and *d.c.b.a* yield the same hash
  - i.e., leaves the hash ambiguous

Example (cont.)

- Tested on connection logs from ICSI’s border
  - July 27 2006
  - 6.2 million connections
- 11% of the connections hash to a unique value
  - i.e., lots of collisions

Example (cont.)



Finding Botnets

- Detectives can combine responses from many witnesses to form a plausible picture of the given botnet

- Detectives then can report results to an aggregater of some form
  - a simple web-based database others can query
  - a behavioral database
  - etc.

- Firewalls and NIDS can then use the information in implementing policy

Finding Botnets (cont.)

- Key open questions:
  - how to form patterns
    - want patterns that are not overly loose or overly tight
  - how to form the loose private matching hashes
    - something easy like multiplication?
    - something more air-tight like private stream matching?
  - defining a flexible, yet computationally tractable query scheme

Summary

- We leverage a wealth of existing technology to sketch two strawman systems for practical sharing of network security-related information

- Designed to respect the tension between the desire to share useful security information and user's and company's privacy issues

- A number of key open research issues

- Key open question: will operators buy in?

References

- Mark Allman, Ethan Blanton, Vern Paxson. *An Architecture for Developing Behavioral History*. USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet, July 2005.
- Mark Allman, Ethan Blanton, Vern Paxson, Scott Shenker. *Fighting Coordinated Attackers with Cross-Organizational Information Sharing*. Under submission.

- Contact:

  `mallman@icir.org`
  `http://www.icir.org/mallman/`