Poster: "Quilt: A system for distributed queries of security-relevant data"

Dr. Timothy Shimeall and George M. Jones and Derrick H. Karimi
CERT/Software Engineering Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213-2612
Contact: http://www.sei.cmu.edu/about/people/

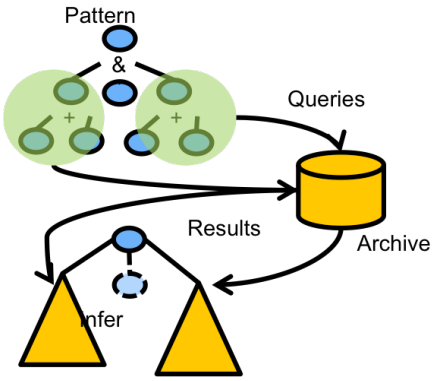

Fig. 1. Quilt Overview

I. Key Contribution

Quilt defines a framework for for automated query, exploration, and analysis of network activity across large scale, real-world diverse data sources in a distributed setting, incorporating the context gained from each data source in the query for the other data sources. It incorporates time as a primary component of the query language, and also supports abductive matching to facilitate identification of patterns where some data may be missing data .

The framework also allows patterns of network behavior to be flexibly defined, which should permit sharing of such patterns between analytical teams, without exposing the underlying data.

II. Problem Being Addressed

Detection of current security threats using existing data sources and methods is difficult. Detection often involves:

- detection of complex, multi-step behaviors across time;
- multiple data sets (IDS alerts, DNS, pcap, flow, etc);
- data distributed across a variety of systems;
- data distributed across administrative domains;
- differing access methods;
- large data that is slow to query;
- large data that is uneconomic to pull back to a central repository;
- missing data due to incomplete collection or intruder actions;
- detection of sequences of events;
- recognition of time/overlap of events;
- detection of patterns involving multiple services, protocols, etc.;
- manual integration across multiple data sources which can be error prone, labor intensive;
- reliance on signature-based alerts;
- limited or primitive capability to bridge multiple data sources to perform coordinated detection.

In addition, signature-based solutions often have the following problems:

- privacy concerns;
- failure to detect innovative attacks;
- failure to detect targeted attacks.

III. Why Is This Problem Interesting Or Important?

- The threats are real and often exhibit temporal or sequential behaviors.
- Operators in this field lack solutions to the problems listed above.
- Selected researchers have access to large quantities of diverse data.
- There appears to be no widely available system that:
  - Permits efficient, effective ways to express queries that build context across data sources;
  - Works at scale on diverse sources of real data;

IV. Approach

A. Design of a query language

The following is a sketch of the Quilt query language grammar. It is conceptually related to STIX and CybOX [1]. The key additions to that structure are the temporal operators at the top of the grammar, and the interpretation of fields as attributes of sources rather than to portions of data structures.

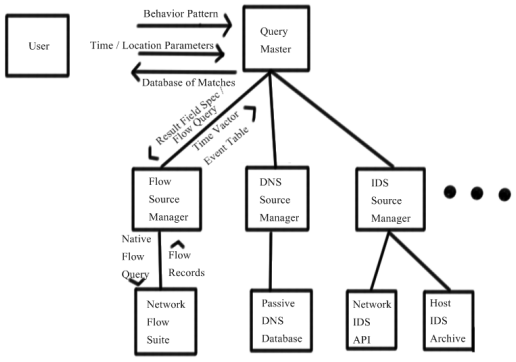

Fig. 2. Quilt Architecture

```
Pattern ::= UNTIL(pattern,pattern) |
            CONCURRENT(pattern,pattern,...) |
            FOLLOWS(dt, pattern, pattern) |
            expr
Expr ::= AND(expr, expr, ...) |
         OR (expr, expr, ...) |
         NOT(expr) | condition
Condition ::= Term > Term | Term == Term |
              Term < Term | Term >= Term |
              Term != Term | Term <= Term |
              Term
Term ::= Term ^ Term | Term * Term |
         Term / Term | Factor
Factor ::= Factor + Factor |
           Factor - Factor | Value
Value ::= ( pattern ) | literal |
          source.field
literal ::= string | numeric
source ::= identifier
field ::= identifier | identifier [ term ]
```

B. Implement a prototype

We are developing a distributed system for implementing temporal, sequential and abductive queries across diverse data sets. The initial system will operate on data from SiLK (NetFlow) [2], Snort (IDS Alerts) [3] and extracted DNS data [4]. See Figure 2.

C. Expanded Example: Phishing email with DNS cache poisoning.

1) Description: Email is sent to all engineering staff at XYZ.com, fraudulently sourced from CIO, requiring participation in security survey at DoSurvey.com (with very short turn-around demanded)

- DNS cache poisoning of DoSurvey at XYZ, redirection to dynamic DNS domain
- Survey asks for "free registration" (email address and user-specified password)
- Survey questions on what network detection is present at XYZ, which servers are used most often
- Users are told responses would enable drawing for cash prize

People often reuse usernames and passwords. The phishers may be counting on this. One further indicator of compromise would be subsequent password attacks.

2) Steps:

action	condition	data needed
email received	*containing* a URL	- IDS Alert (phishing)
	while DNS poisoned for domain	- DNS data - Blocklists
	followed by	
web hit on Phishing blocklist		- Flow data
	followed by	
brute force		- System Logs

3) Query Processing:

- a Quilt *sourceManager* tracks each of several sources: DNS, Network Flow, Blocklists, email server logs, IDS alerts.
- An IDS alert fires indicating possible phishing. The alert indicates time and suspect URL.
- A user process formulates a *quiltQuery* specifying the URL and the time the message was received.
- The *queryMaster* decomposes the *quiltQuery* into a series of *sourceQueries*
- The first *sourceQuery*, sent to the *DNS sourceManager* asks "was this DNS name poisoned at the time the mail was received"
- The second *sourceQuery*, sent to a *Netflow sourceManager* asks "did we see web traffic to the poisoned address following receipt of the phishing email"?
- If the answer both answers are "yes", then return a match to the *quiltQuery* indicating a successful phishing attempt.

REFERENCES

[1] S. Barnum, "Standardizing cyber threat intelligence information with the structured threat information expression (stix)," *MITRE Corporation, July*, 2012.

[2] C. Gates, M. Collins, M. Duggan, A. Kompanek, and M. Thomas, "More netflow tools for performance and security," in *Proceedings of the 18th USENIX conference on System administration*. USENIX Association, 2004, pp. 121–132.

[3] M. Roesch *et al.*, "Snort-lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX conference on System administration*. Seattle, Washington, 1999, pp. 229–238.

[4] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer networks*, vol. 31, no. 23, pp. 2435–2463, 1999.