Taking Event Correlation With You

Rob King

Black Hat Briefings 2015

$ who am i

- Rob King <jking@deadpixi.com> (yes "j" not "r" — it's a first/middle name thing)
- Security researcher and programmer
- Heavy focus on system architecture, traffic analysis, and event correlation

The Goal of This Presentation

"Leveraging" makes me sound like I understand business stuff. I don't.

Introducting the Giles Production Rule System Compiler, a tool that makes it easy...

- ...to build event correlation engines.
- ...to find patterns defined by complex predicates in large, constantly changing datasets.
- ...to integrate these sorts of systems into your existing environment, leveraging existing programmer expertise.

A Brief History of Giles

$+ \quad = \text{DIRT} \rightarrow \text{Giles}$

Just the Facts, Ma'am

What is an Event/Fact?

A *fact* is a statement that we know to be true within our problem domain.

- Suzanne is an astronaut.
- Train $\sqrt{-1}$ arrived at platform $9\frac{3}{4}$ at 26:00.
- The user's preferred language is Canadian French.
- User `jdoe` logged into terminal #33 at 17:10.

Giles calls them *facts* instead of *events* because Giles can create more than just event correlation engines.

A Collection of Facts

All Facts in the System are Called the *Working Memory*

Contents of `/var/log/logins`
LOGIN OK: jdoe 17:10 term 92
LOGIN OK: epalpatine 17:33 term 66
LOGIN FAILED: ffoobar 17:34 term 81
LOGIN OK: ffoobar 17:34 term 81

Imposing Structure on Facts

Facts and Fields

Contents of `/var/log/logins`			
Status	**Username**	**Login Time**	**Terminal**
LOGIN OK:	jdoe	17:10	term 92
LOGIN OK:	epalpatine	17:33	term 66
LOGIN FAILED:	ffoobar	17:34	term 81
LOGIN OK:	ffoobar	17:34	term 81

Imposing Types on Fields

Field Name	**Field Type**
Login Successful	`BOOLEAN`
Username	`STRING`
Login Time	`INTEGER`
Login Terminal	`STRING`

Extracting Knowledge From Facts

Detecting Bad Logins

- `grep` works here
- No state to be captured

Contents of `/var/log/logins`			
Status	**Username**	**Login Time**	**Terminal**
LOGIN OK:	jdoe	17:10	term 92
LOGIN OK:	epalpatine	17:33	term 66
LOGIN FAILED:	ffoobar	17:34	term 81
LOGIN OK:	ffoobar	17:34	term 81

Extracting Knowledge From Facts

Detecting Multiple Failed Logins Within a Time Window

- Some state needs to be stored: the user name and the start time

Contents of `/var/log/logins`

Status	Username	Login Time	Terminal
LOGIN OK:	epalpatine	17:33	term 66
LOGIN FAILED:	ffoobar	17:34	term 81
LOGIN FAILED:	ffoobar	17:34	term 81
LOGIN OK:	lcalrissian	17:34	term 80
LOGIN FAILED:	ffoobar	17:34	term 81

Some questions to consider:

- Whose clock do we use for timing? The wall, or the login records'?
- When do we reset the timer?

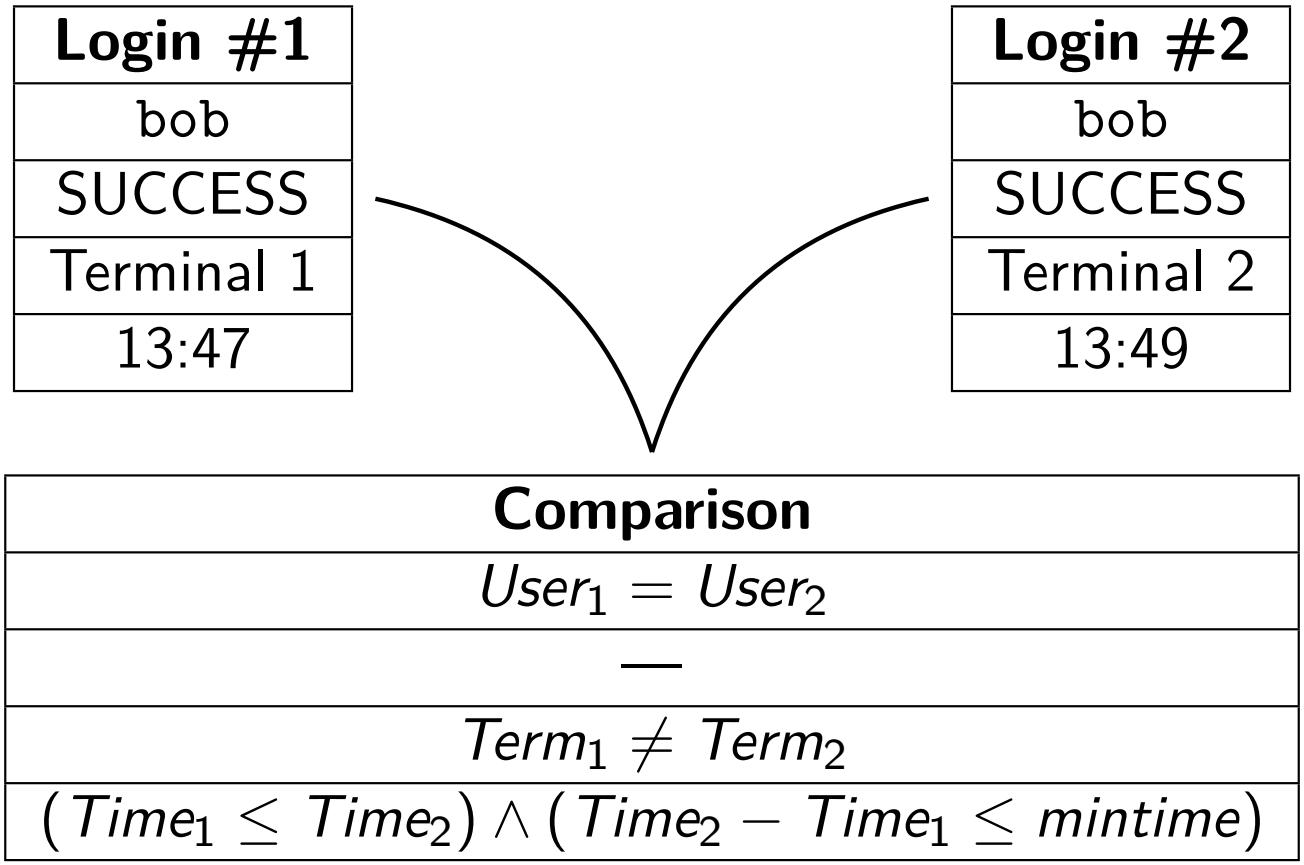
Extracting Knowledge From Facts

Complex Event Correlation (A Word Problem Because Rob Can't Draw Well With TikZ)

- Detect arbitrary relationships between sets of facts.
- Example: Detecting a compromised password by seeing multiple logins:

  If Bob logs in to Terminal 1 at 13:47, and logs in to Terminal 2 at 13:49, and we know that the two terminals are geographically far apart and don't allow remote logins...
  ...then we know that someone is impersonating Bob in at least one of those logins.

Detecting This Scenario



Questions About Complex Correlation

- How do we efficiently detect *all* impersonating logins?
- How do we handle out-of-order reporting?
- How do we handle retraction of facts?
- How do we scale to thousands or millions of facts?

Introducing Giles

Finally

Giles is an open source compiler that turns a normal, standalone database into a powerful event correlation engine.
Giles engines can...

- be treated as normal databases with no external dependences, *because they are normal databases*
- handle out-of-order assertions of facts
- scale to millions of facts
- handle fact retraction consistently
- reveal their reasoning process (fact provenance and justification)
- run efficiently in time

Live Demo

Let's take a look at a real Giles engine...

Advantages of Giles

- You can take event correlation with you: anywhere you can run SQLite!
- Your programming team can use existing, ubiquitous database APIs and expertise to perform event correlation.
- Your engines scale as well as your database, and lots of energy has been put into making databases scalable.
- Your engines are generally safe the in face of crashes and outages.
- Your engines can have transactional semantics.
- The tool itself is open source and commercial-use-friendly.

Uses for Giles Engines

- Forensics
  - Record Linkage
  - Post-hoc Behavior Analysis
  - Pentesting Expert Systems
- Operations
  - Log Analysis
  - Event Correlation
- Business Logic and Pattern Matching
  - Decision Systems
  - Diagnostic Systems

Performance

In Theory — One Can Write Slow Code in Any Language!

- Adding n facts to a "regular" database that already contains t facts and then querying for matches: $O(t + n)$ time.
- Adding n facts to a Giles engine that already contains t facts and then querying for matches: $O(n)$ time.

Under the Hood

The Rete Algorithm

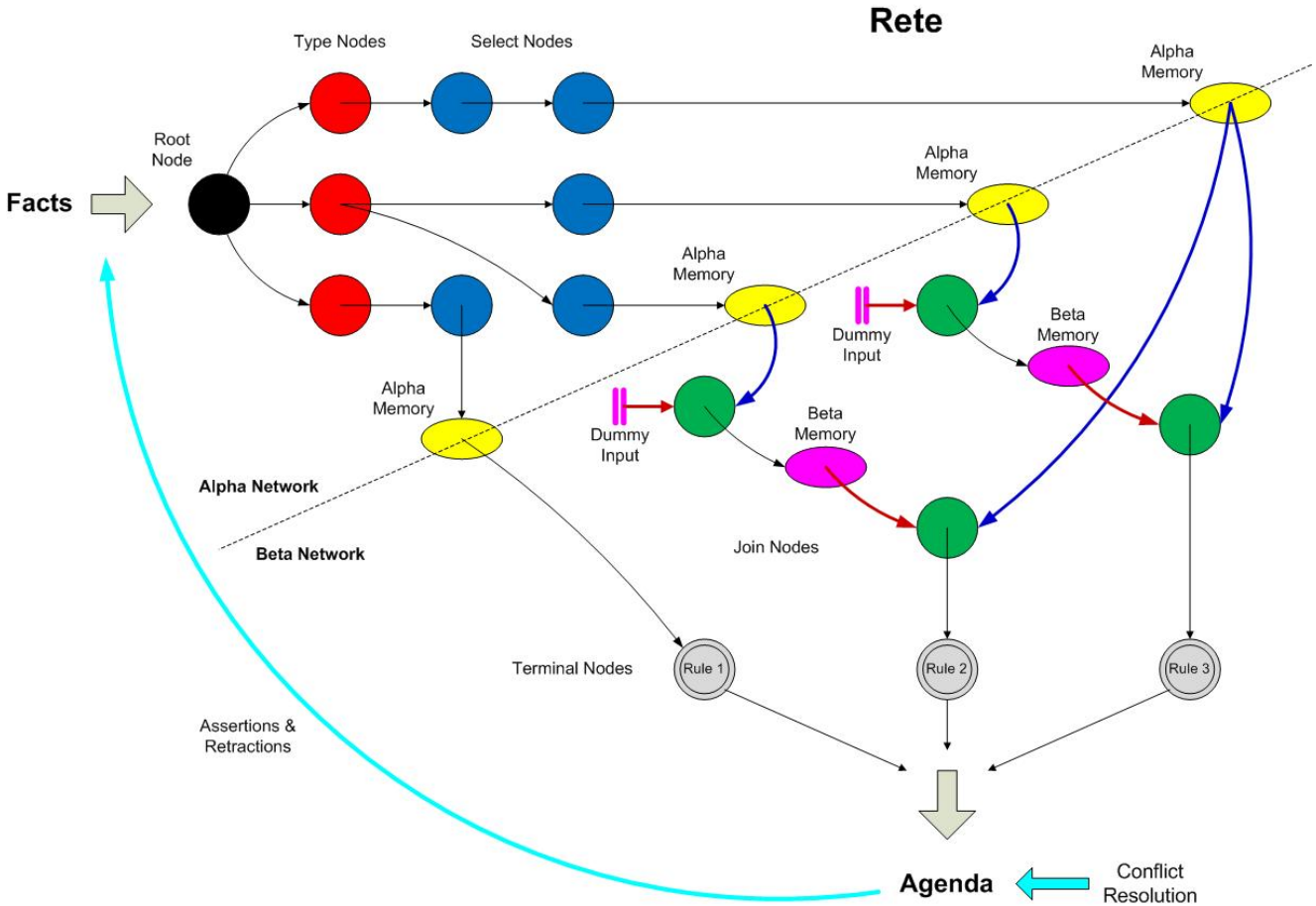


Figure: Courtesy Wikimedia Commons

Rete and SQL

- α- and β-memories are tables
- Triggers fire when a memory is changed to update the contents of other memories or assert/retract facts
- α-predicates become triggers that suppress `INSERT`s on α-memory tables
- Expressions are rewritten to be more amenable to query optimization
- A set of indexes is computed to provide maximal coverage to all generated queries

Black Hat Sound Bytes

- Complex event correlation engines have great applications in writing security (and secure!) software. Complex event correlation engines don't have to be complex, though!
- Giles engines are just normal databases, meaning they can use existing APIs and programmer expertise, and take advantage of modern databases' performance and safety features.
- I'm reachable at jking@deadpixi.com, and Giles is available at `https://git.korelogic.com/giles.git`.