Defense in Depth: In Depth

Presented by: Chelsea H. Komlo

About me

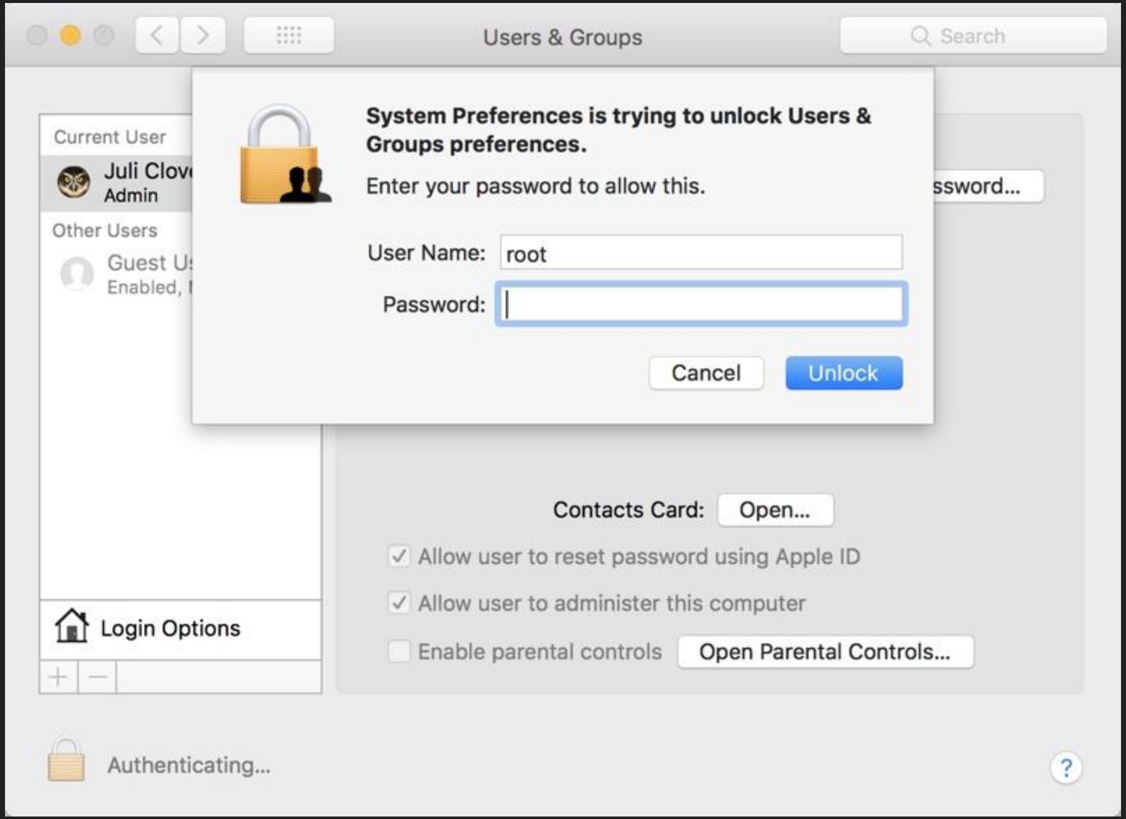
- Software engineer, privacy and security engineer
- HashiCorp, ThoughtWorks, Tor
- Worked in 5 countries and two languages

About this talk

- NOT how to do security
- The purpose of this talk to discuss how to think defensively about your system at every level.

What I often come across when talking about security

You could have the most awesome encryption standard, but pressing the enter key could sidestep all authentication.



System Preferences is trying to unlock Users & Groups preferences.

Enter your password to allow this.

User Name: root

Password:

Contacts Card: Open...

✓ Allow user to reset password using Apple ID

✓ Allow user to administer this computer

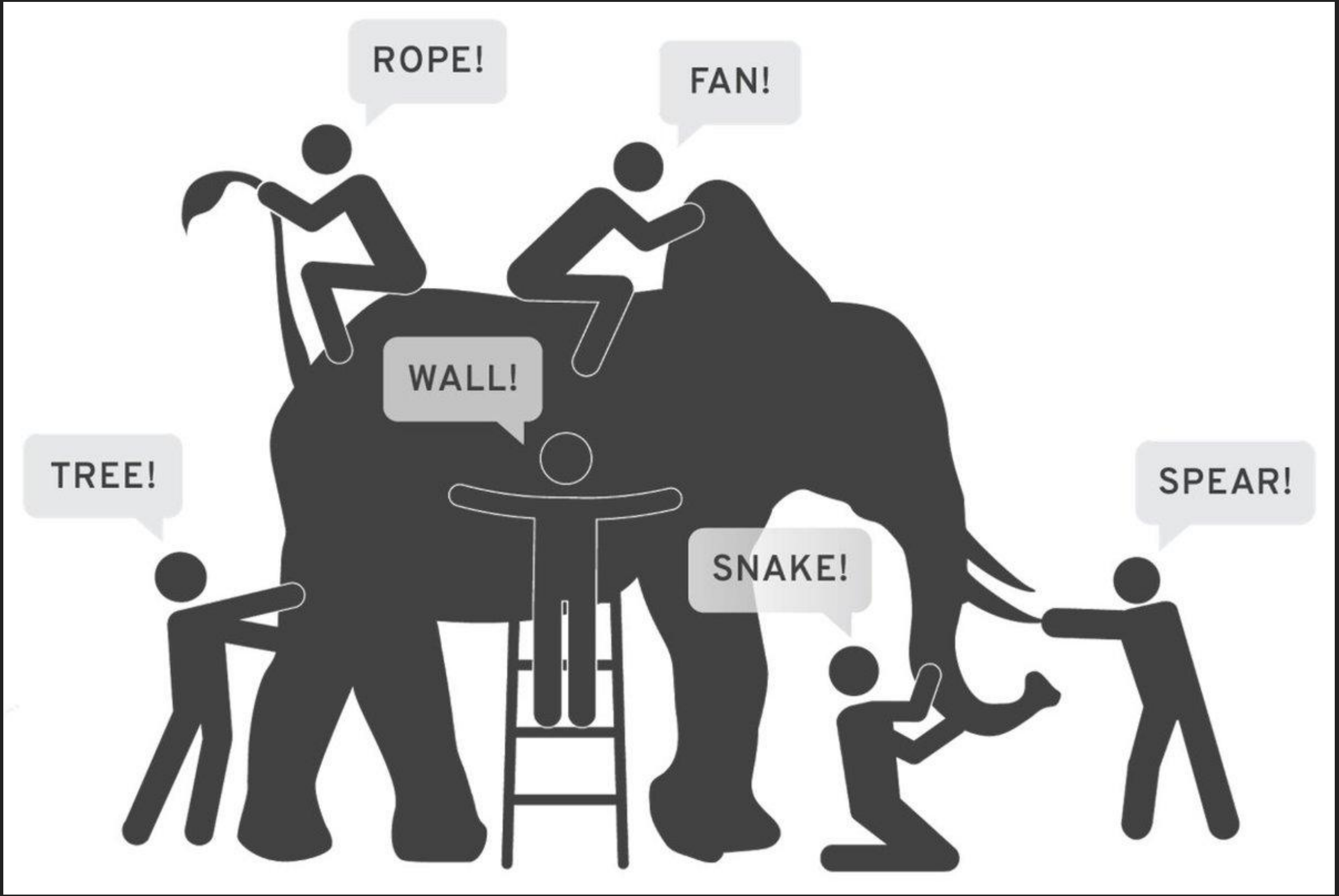
☐ Enable parental controls

Authenticating...

One vulnerable third-party library leads to hundreds of millions of sensitive PII being stolen

EQUIFAX®

Security is holistic.



Defense in depth is necessary for a secure system

Goal: One vulnerability won’t result in compromising the entire system.

We’ll look at defense in depth from a variety of viewpoints

- Low level (code)
- Mid level (teams)
- High level (architecture)
- Highest level (product strategy)

Defense in depth: Code

- Maintain code quality
- Leverage automated tooling
- Meaningful automated tests

Defense in Depth: Maintain code quality

- Antipattern: Making assumptions when writing code.
- Pattern: Code should written defensively
- Takeaway: Security vulnerabilities are bugs!

Example: Brittle code

```
// Should never be called with nil
func sayName(p *Person) {
  fmt.Printf(“%s”, p.Name)
}
```

Defense in Depth: Leverage automated tooling

- Antipattern: Minimal compile-time validation
- Pattern: Enable language-specific compile-time checks
- Takeaway: Humans fail! Leverage automated tooling where possible

Example: Automated code analysis

- Go Race Detector
- ASAN
- GCC: -Wall -Wextra

Defense in Depth: Meaningful automated test cases

- Antipattern: Adding a single test case for a function
- Pattern: Having test cases that exercise your code with varying granularity.
- Takeaway: Don’t be single-dimensional in your tests!

Testing at multiple levels:

- Unit
- Integration
- E2E
- Soak
- Time-based
- Fuzzing

Defense in depth: Teams

- No more “rock stars”
- No “throw over the wall” security requirements

Defense in Depth: No more rock stars

- Antipattern: Someone on the team pushing lots of code to master without a review.
- Pattern: All code goes through thorough code review (from anyone on the team)
- Takeaway: Security is a team sport!

Defense in Depth: No “throw over the wall” security requirements

- Antipattern: Long list of requirements from your security team.
- Pattern: Development teams and security teams closely collaborating.
- Takeaway: Collaborate.

Defense in depth: Architecture

- Managing evolution cleanly
- Automate infrastructure

Defense in Depth: Manage evolution cleanly

- Anitipattern: Layers of “cruft” and deprecated features.
- Pattern: Remove deprecated code paths, strive for minimal branching.
- Takeaway: Your attacker will know your system better than you will!

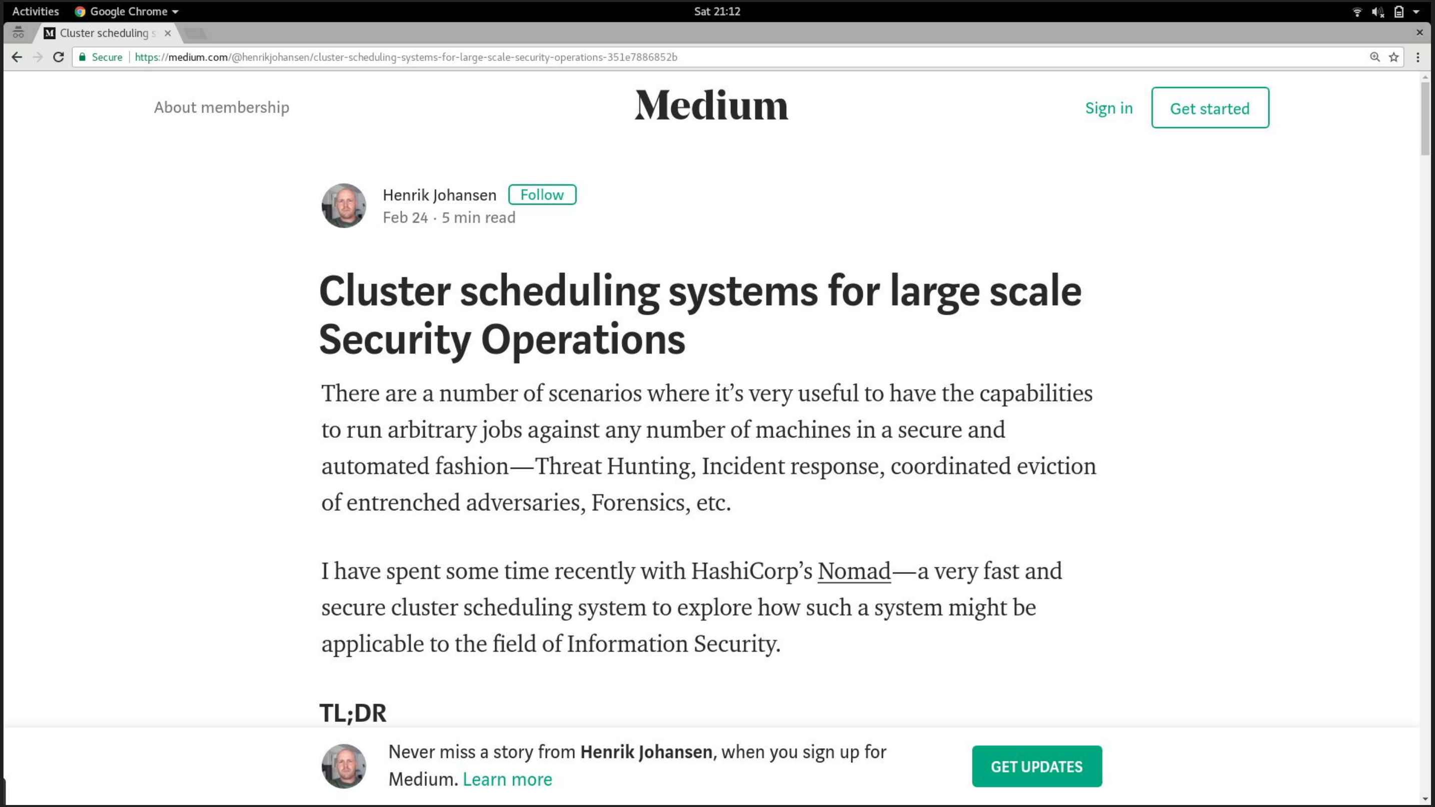
Example: OpenSSL versus OpenBSD’s LibreSSL

Over 90,000 lines of code removed.

Defense in depth: Automate infrastructure

- Anitipattern: Bespoke, artisanal server management.
- Pattern: Use automated tooling to manage your cluster.
- Takeaway: The less manual effort, the fewer “forgotten holes.”

Example: Cluster schedulers for Secops


Activities
Google Chrome
Sat 21:12
Cluster scheduling s
Secure
https://medium.com/@henrikjohansen/cluster-scheduling-systems-for-large-scale-security-operations-351e7886852b
About membership
Medium
Sign in
Get started
Follow

Henrik Johansen
Feb 24 · 5 min read

Cluster scheduling systems for large scale Security Operations

There are a number of scenarios where it's very useful to have the capabilities to run arbitrary jobs against any number of machines in a secure and automated fashion—Threat Hunting, Incident response, coordinated eviction of entrenched adversaries, Forensics, etc.

I have spent some time recently with HashiCorp's Nomad—a very fast and secure cluster scheduling system to explore how such a system might be applicable to the field of Information Security.

TL;DR

Never miss a story from **Henrik Johansen**, when you sign up for Medium. Learn more

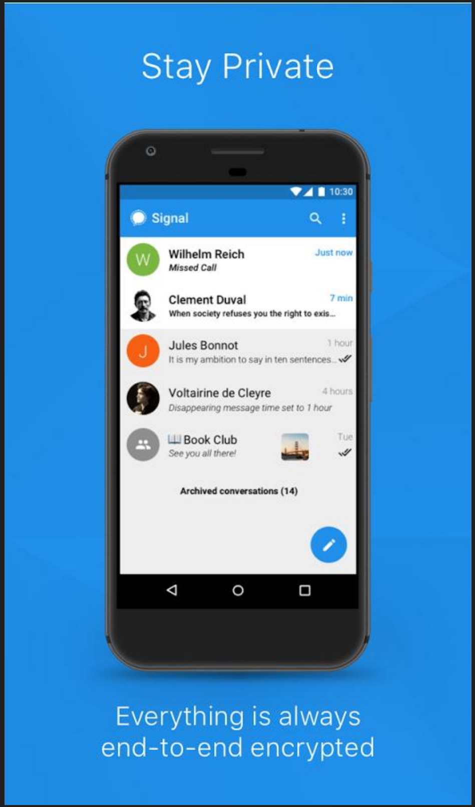
Defense in depth: Product Strategy

- Privacy and security serve the same ends
- Consider your users’ threat model

Defense in Depth: Privacy and security serve the same ends

- Antipattern: Collecting all possible data
- Pattern: Collect only what is strictly necessary
- Takeaway: Strive for privacy by design, as opposed to retroactive privacy.

Example: Encrypted messaging applications



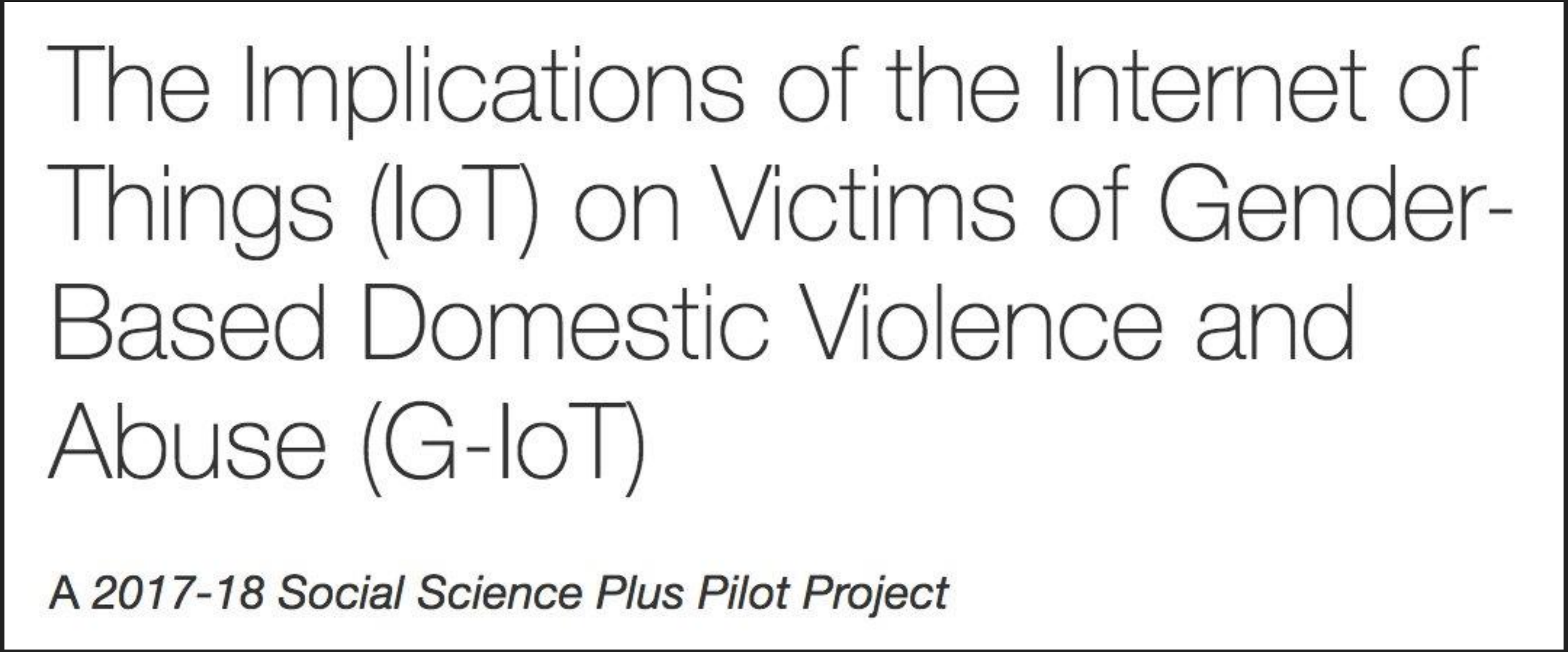
Defense in Depth: Consider your users’ threat model

- Antipattern: Planning for only your organization’s security needs
- Pattern: Consider every user’s needs, including at-risk users in your threat model
- Takeaway: Be aware of decisions that place users at greater risk

Example: Sensitive data and third parties

Cambridge Analytica
The Power of Big Data and Psychographics in the Electoral Process

Example: Consider vulnerable users



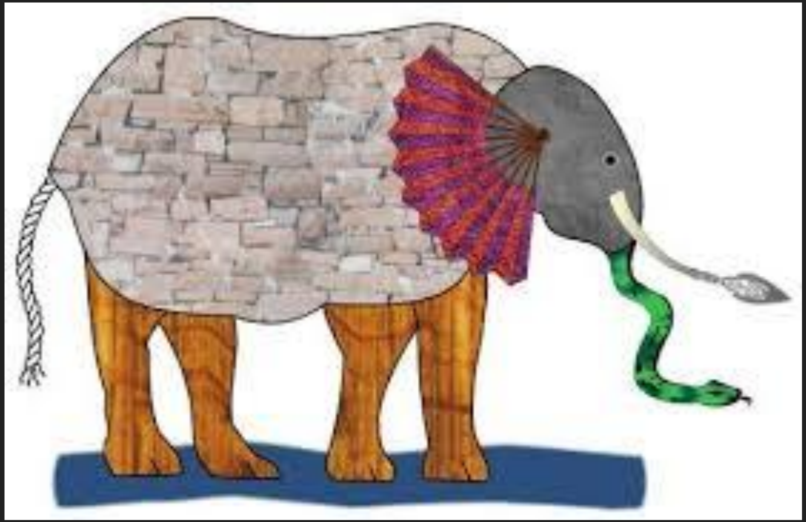
The Implications of the Internet of Things (IoT) on Victims of Gender-Based Domestic Violence and Abuse (G-IoT)

A 2017-18 Social Science Plus Pilot Project

Security must be holistic!

This means all roles, all people, working together thoughtfully.

There is no partial credit in security!



Thank you!