Access Control

COS 316

Why might we want to control access to resources?

The Complete Guide to Facebook Privacy

Despite repeated privacy lapses, Facebook offers a fairly robust set of tools to control who knows what about you.

Facebook has never been particularly good at prioritizing your privacy. Your data powers its business, after all. But recent revelations that a firm called Cambridge Analytica harvested the personal information of 50 million unwitting Facebook users in 2015 has created new sense of urgency for those hoping for some modicum of control over their online life. If you ever needed a wake-up call, this is it.

The good news: Despite the repeated, public privacy lapses, Facebook does offer a fairly robust set of tools to control who knows what about you—both on the platform and around the web. The bad news: Facebook doesn't always make those settings easy to find, and they may not all offer the level of protection you want.

Fear not! Below, we'll walk you through the steps you need to take to keep advertisers, third-party apps, strangers, and Facebook itself at bay. And if after all that you still feel overly exposed? We'll show you how ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE STORE

POLICY —

Meet the e-voting machine so easy to hack, it will take your breath away

Virginia decertifies device that used weak passwords and wasn't updated in 10 years.

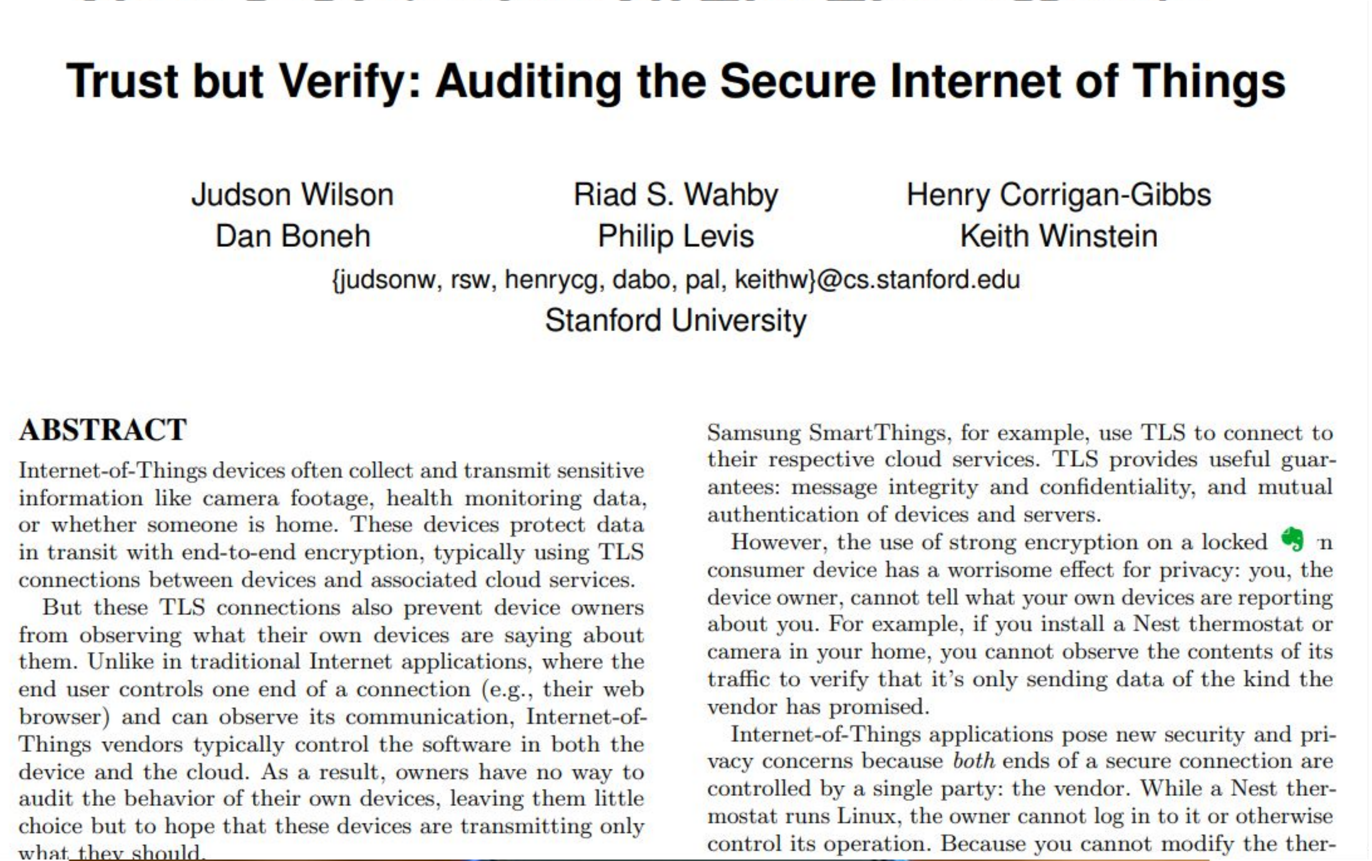
DAN GOODIN - 4/15/2015, 2:55 PM

VerifiedVoting.org

Enlarge

Why might we *not* want to control access to resources?

177,895 views | Apr 19, 2019, 08:32pm



Trust but Verify: Auditing the Secure Internet of Things

Judson Wilson, Riad S. Wahby, Henry Corrigan-Gibbs
Dan Boneh, Philip Levis, Keith Winstein
{judsonw, rsw, henrycg, dabo, pal, keithw}@cs.stanford.edu
Stanford University

ABSTRACT

Internet-of-Things devices often collect and transmit sensitive information like camera footage, health monitoring data, or whether someone is home. These devices protect data in transit with end-to-end encryption, typically using TLS connections between devices and associated cloud services.

But these TLS connections also prevent device owners from observing what their own devices are saying about them. Unlike in traditional Internet applications, where the end user controls one end of a connection (e.g., their web browser) and can observe its communication, Internet-of-Things vendors typically control the software in both the device and the cloud. As a result, owners have no way to audit the behavior of their own devices, leaving them little choice but to hope that these devices are transmitting only what they should.

Samsung SmartThings, for example, use TLS to connect to their respective cloud services. TLS provides useful guarantees: message integrity and confidentiality, and mutual authentication of devices and servers.

However, the use of strong encryption on a locked n consumer device has a worrisome effect for privacy: you, the device owner, cannot tell what your own devices are reporting about you. For example, if you install a Nest thermostat or camera in your home, you cannot observe the contents of its traffic to verify that it's only sending data of the kind the vendor has promised.

Internet-of-Things applications pose new security and privacy concerns because *both* ends of a secure connection are controlled by a single party: the vendor. While a Nest thermostat runs Linux, the owner cannot log in to it or otherwise control its operation. Because you cannot modify the ther-

A (slightly) formal model

- Objects: the things being accessed
    - A file, database table, network socket, satellite imagery of “nuclear facilities,” missile launcher...
- Subjects: an entity that requests access to an object
    - A process, network endpoint, etc…
    - **Principal**: some unique a account or role, such as a user
- Authentication: a proof that a subject speaks for some principal
    - E.g. logging in with a username & password
- Authorization: the particular rules that govern subjects’ access to objects
- Secrecy: who might learn the contents of an object
- Integrity: who may have influenced the contents of an object

Ad-hoc access control

- Access policy enforcement is scattered throughout system

```
fn (profile *Profile) viewProfile(user) (HTML) {
  if profile.public ||
     profile.friends.contains(user) {
    return profile.HTML
  } else {
    return HTML.Forbidden
  }
}
```

```
fn (profile *Profile) viewFullName(user) (HTML) {
  if profile.public || user.handle ==
                               “NSA_Backdoor” {
    return profile.FullName.HTML
  } else {
    return HTML.Forbidden
  }
}
```

- Very common in applications with lots of users. Why?

Ad-hoc access control

- Application-specific access rules
- Data for rules stored separately from data objects
  - Really a problem of granularity

Profile Table

id	**full_name**	**profile_pic**	**handle**	**bio**
1	Amit Levy	/i/1f3.png	aalevy	Dog dad, foodie, yog...
2	Alan Kaplan	/i/a60.png	kap	Enjoys long function names...

Friends Table

follower	**followee**
1	2
2	1
1	4
1	5
...	...

Problems Ad-hoc access control

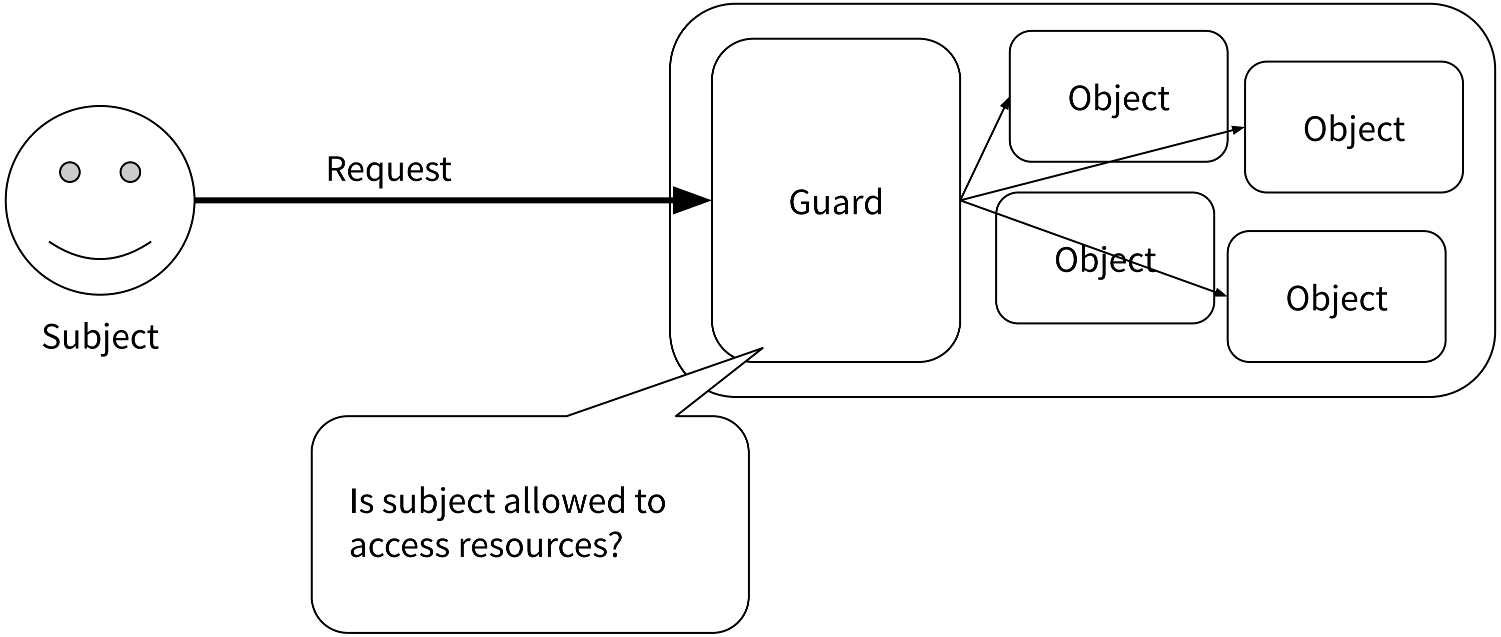
- Policy is emergent

```
fn (profile *Profile) viewProfile(user) (HTML) {
  if profile.public ||
     profile.friends.contains(user) {
    return profile.HTML
  } else {
    return HTML.Forbidden
  }
}
```

```
fn (profile *Profile) viewFullName(user) (HTML) {
  if profile.public || user.handle ==
                              “NSA_Backdoor” {
    return profile.FullName.HTML
  } else {
    return HTML.Forbidden
  }
}
```

- Who can view a user’s full name?

The Guard Model



Examples of the Guard Model

- Kernel
    - File system permissions: as long as objects modeled as files, access checks are centralized
    - Reference monitor
- Networks
    - Firewall
    - Apache HTTP server's `.htaccess` rules
- Databases
    - Table/database visibility
    - Limit ability to `ALTER`, `UPDATE`, `DROP`, etc

The Guard Model

A mechanism, leaves us with many questions:

- What kinds of rules does the guard enforce?
- Who gets to set or change the rules?
- What is the granularity of subjects and objects?
- Who gets to create new principals?

Answers to these questions help determine the expressivity, performance, and security of the system.

What kinds of rules?

There are many “policy languages”

- Access control lists: which subjects can read/write which objects
- Capabilities: unforgeable tokens that encode specific rules on objects
  - Subjects unnamed
- Information flow: the relationship between data sources and data sinks
  - Neither subjects nor objects named, instead

Who sets the rules?

We will discuss two broad categories:

- Discretionary Access Control (DAC)
    - Very common, e.g. UNIX user/group permissions
- Mandatory Access Control (MAC)
    - Pretty uncommon, much more robust
    - E.g. SE-Linux & AppArmore, and lots of research systems

Granularity

Why doesn't database just re-use UNIX file permissions?

- The objects in UNIX file permissions are *files*, with read/write/execute permissions
- But...
- Tables & schemas might span many files
- Databases might include several schemas or tables in a single file
- Alter, update, drop don't map well to read/write/execute
  - E.g. UPDATE should retain layout of data in a file

Granularity

Why doesn't web application re-use database permissions

Profile Table

id	full_name	profile_pic	handle	bio
1	Amit Levy	/i/1f3.png	aalevy	Dog dad, foodie, yog...
2	Alan Kaplan	/i/a60.png	kap	Enjoys long function names...

Friends Table

follower	followee
1	2
2	1
1	4
1	5
...	...

Centralized vs. Decentralized Access Control

Why don’t web applications re-use UNIX users/groups?

- Facebook does not have a UNIX user for you on their servers. Why?
- UNIX does not allow unprivileged users to create new principals
- Web applications run as a single UNIX user, and re-implement:
    - Authentication
    - Authorization
    - Guard
    - ...

Summary

- Access control is a reflection of some real-world policy
  - Design with care
- Ad-hoc access control is common, but problematic, so we want *systems*
- The guard model helps separate security enforcement from other functionality
- Behavior of security system determined by:
  - Policy rules
  - Granularity of subjects/objects
  - Mandatory vs. Discretionary
  - Centralized vs. Decentralized Principals