ACLs Don’t

Justin Cormack reads Tyler Close

ACLs Don’t by Tyler Close

- HP Labs Technical report, 2009
- http://www.hpl.hp.com/techreports/2009/HPL-2009-20.html
- Submitted to IEEE Symposium on Security and Privacy, but not apparently accepted

Why this paper?

- model of clarity, short and readable
- makes you think about a problem in a different way
- synthesis of a lot of thought
- underground cult paper, now you too can join the cult
- access control is a hugely important problem
- lesson still not understood!

part one: the paper

In the last few years, increasing attention has been devoted to attacks which are distinctly different in nature from the major vulnerabilities discussed in the past. Previously, buffer overflow, SQL injection and Cross-Site Scripting (XSS) garnered the most attention. These attacks all share a common modus operandi in that they inject code into a victim program and thus make it behave according to the attacker's wishes.

Recent attacks such as Cross-Site Request Forgery (CSRF) and clickjacking don't fit this familiar mold.

Neither attack requires injecting code chosen by the attacker into the victim program. Instead, both attacks use the victim's existing program logic to unexpected ends. Using messages that follow the syntactic conventions expected for legitimate requests, the attacker makes use of resources that should be inaccessible according to the application's access policy.

access control matrix

Compiler service example

Table 1. Access matrix for file access

	main.c (m)	a.out (a)	log.txt (l)
Vendor (V)			read, write
User (U)	read, write	read, write	

Compiler as a service

Table 2. Access matrix for an instance of the compiler program

	Compiler (c)	main.c (m)	a.out (a)	log.txt (l)
Vendor (V)				read, write
User (U)	call	read, write	read, write	
Compiler (C)		read	write	write

This is a least privilege configuration!

Table 2. Access matrix for an instance of the compiler program

	Compiler (c)	main.c (m)	a.out (a)	log.txt (l)
Vendor (V)				read, write
User (U)	call	read, write	read, write	
Compiler (C)		read	write	write

Running as user has too many permissions, vendor has too few. In practise running least privilege is disappointingly rare… many people will make the compiler “suid” so it can pretend to be multiple users. More later...

two implementations

ACL model

- message consists of an identification of the sending process followed by an arbitrary amount of data.
- the identification is provided by the system and therefore cannot be forged.
- system then checks the ACLs
- user: `Compiler.compile("main.c", "a.out")`
- compiler: `Filesystem.read("main.c")`
- compiler: `Filesystem.append("log.txt", "log entry")`
- compiler: `Filesystem.write("a.out", "compiled code")`

Capability model

- a message consists of a list of permissions and an arbitrary amount of data.
- each permission in the list is selected by the message sender, choosing from its held permissions.
- the selected permissions are added to the message by the system and therefore cannot be forged.
- our most common model of capabilities is file descriptors, although there are other models

Capability model continued

- user: `fd1 = Open("main.c")`
- user: `fd2 = Open("a.out")`
- vendor: `fd3 = Open("log.txt")`
- vendor: `New Compiler(fd3)`
- user: `Compiler.compile(fd1, fd2)`
- compiler: `Filesystem.read(fd1)`
- compiler: `Filesystem.append(fd3, "log entry")`
- compiler: `Filesystem.write(fd2, "compiled code")`

In the real world, eg AWS

- ACL model corresponds to AWS ACL system
- eg create lambda with ACL allowing access to S3 bucket
- capability model corresponds to passing signed URLs that grant access to S3 API calls.
- most systems only support ACL model

confused deputy problem

Confused deputy

`Compiler.compile("main.c", "log.txt")` (send compile output to log file) is allowed in ACL model!

Table 2. Access matrix for an instance of the compiler program

	Compiler (c)	main.c (m)	a.out (a)	log.txt (l)
Vendor (V)				read, write
User (U)	call	read, write	read, write	
Compiler (C)		read	write	write

ACLs don’t authorize correctly

The capability model is still correct

- The passed fd (or object or URL) is checked at creation time (fd) or based on creator’s signature (signing allows check to be deferred)
- The result of an access check is reified as a capability that can be transferred to other principals, and so used in messages that combine the permission with those of the other principals.

When viewed in this way, it is clear that the ACL model and the capability model are fundamentally different ways of modeling the authorization of requests. Although both models can be construed as algorithms operating on an access matrix, they use different parts of the matrix at different times and grow the matrix in different ways, resulting in different access decisions for identical scenarios. 

two parties and multi party are very different

- earlier literature treated both models as different views of access matrix
- this is true if there are only two parties involved

BUT

- there are often more than two parties
- consider microservices, we always have lots of parties!

In addition

- sending a message in an ACL system does not indicate intent and ownership
- the compiler is not accountable for writing to the output file, this was at the user’s request but the permission check references the compiler.
- the access matrix does not explain how permissions got there.
- delegating permissions is important in the real world
- valet parking, an accountant filing your taxes, ...
- we delegate permissions all the time.
- capabilities are just permissions that can be delegated.

examples from the paper

cross-site request forgery

- send a POST to a site you are logged into from another website
- example is to buy something
- the browser is the third party, can act for you, has your cookies
- solution is to require a random token that is only available on the real site, eg a random number per instance of the form
- the random token acts like a capability, unforgeable, provides access

also... just with the browser as the third party

- clickjacking
- click fraud
- ...

part two: into the present

a recent example

Kubernetes CVE-2018-1002105

From the postmortem

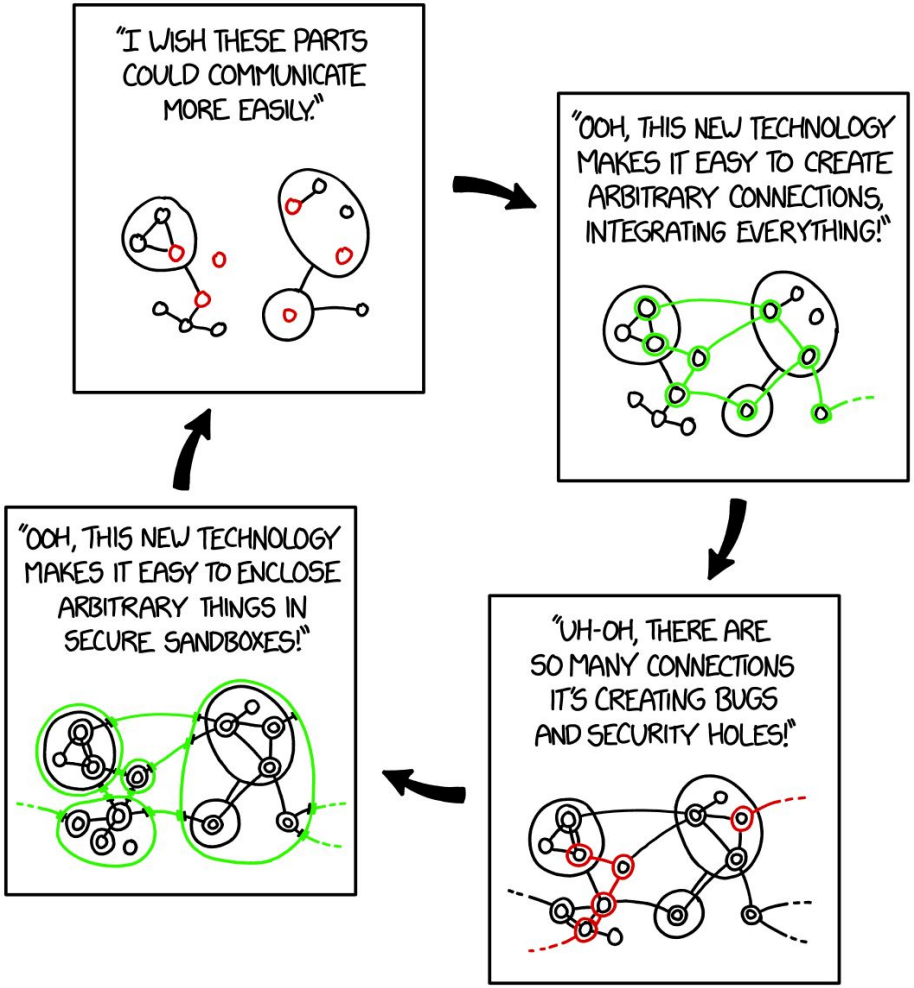
- Kubernetes API server proxy components still use http/1.1 upgrade-based connection tunneling, which does not distinguish between request data sent by the apiserver while establishing the backend connection, and data sent by the requesting user
- High and low-privilege API requests to aggregated API servers are proxied via the same component with the same high-permission transport credentials

A complex mix of issues, with confused deputy part of the mix...

Kubernetes CVE-2018-1002105

- not a least privilege setup, the application server has all privileges and can act as any user
- this makes it even easier for security issues
- transfering capabilities still help in this case rather than the server just having to state who it wants to act as

why has nothing changed?


"I WISH THESE PARTS COULD COMMUNICATE MORE EASILY."
"OOH, THIS NEW TECHNOLOGY MAKES IT EASY TO CREATE ARBITRARY CONNECTIONS, INTEGRATING EVERYTHING!"
"UH-OH, THERE ARE SO MANY CONNECTIONS IT'S CREATING BUGS AND SECURITY HOLES!"
"OOH, THIS NEW TECHNOLOGY MAKES IT EASY TO ENCLOSE ARBITRARY THINGS IN SECURE SANDBOXES!"

A decade since the paper, no real change?

- RBAC is still what everyone does
- the browser has mitigated issues but they appear elsewhere
- microservices, serverless, containers all have many parties, so the problem is becoming more severe

what exactly is a capability?

- unforgeable, transferable authority
- it can “be” an object or an object reference or a safe pointer locally
- locally can enforce with OS (fd) or language (object)
- if you have it you can use it, you cannot create one from scratch
- in distributed world this doesn’t exist...

good work on capabilities locally

- Capsicum in FreeBSD extends file descriptors and how they can be restricted, adds process descriptors
- ECMAScript spec proposal for Realms API allows least authority object capabilities in JavaScript
- a capability is just an unforgeable object reference, within programming languages capabilities can be enforced by the type system, in the absence of global variables
- no ambient authority - most languages have lots of ambient authority, you can just import anything

some availability of distributed capabilities

- mechanisms like API request signing on AWS, but somewhat ad hoc
- JWT is a capability system because it is a bearer token, although it is rarely used as one, more because it can pass through TLS termination; for the same reason it can be passed by a deputy.
- generally used not as capabilities but as login tokens and not supposed to be passed on.
- lack of design patterns and documentations for using JWT as capabilities effectively.

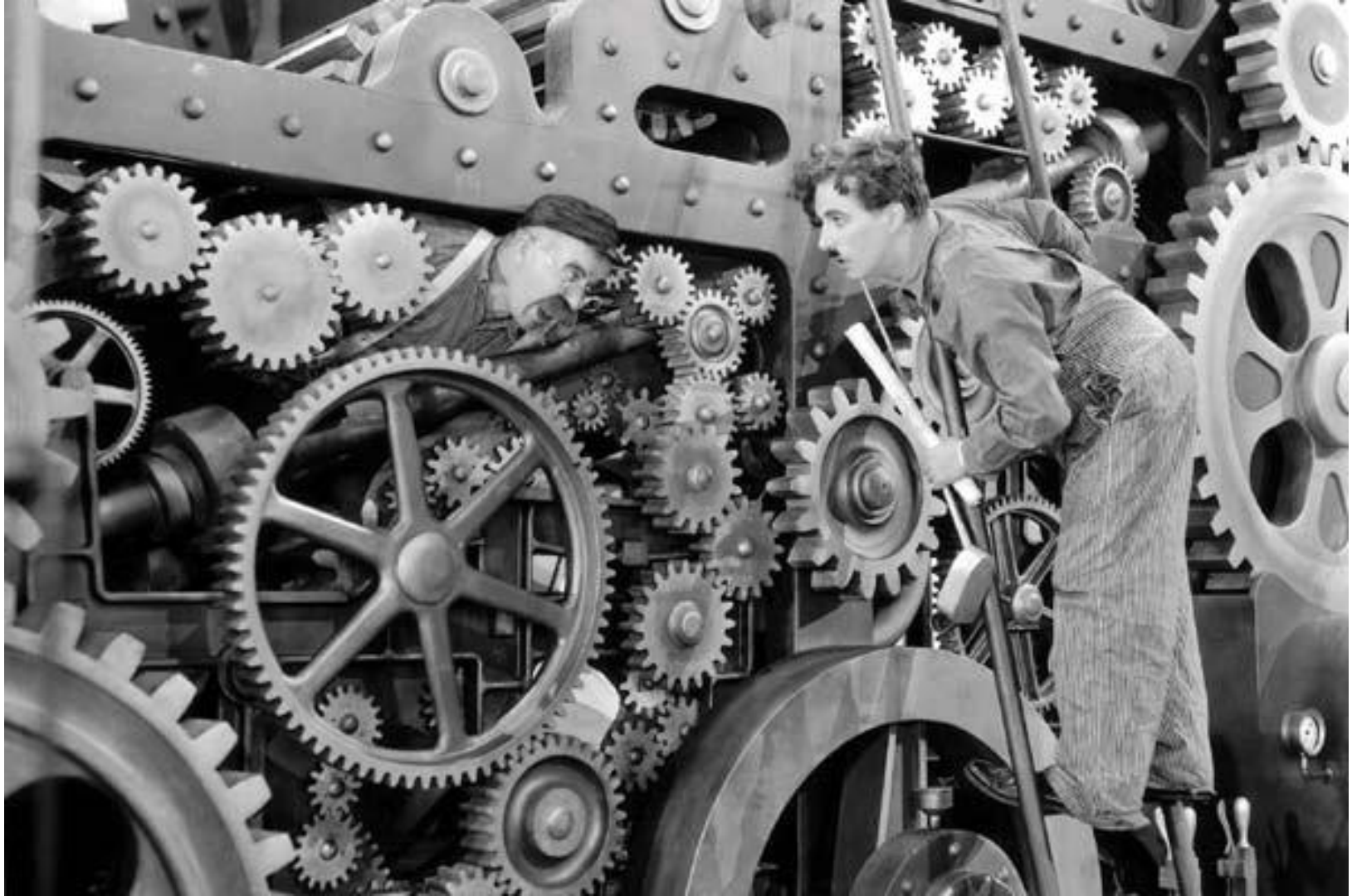
capability mechanisms

- “Swiss numbers” just create a long random number
- signatures eg JWT, AWS signed URLs
- macaroons, a great paper *Macaroons: Cookies with Contextual Caveats*
- asymmetric encryption (something I have been working on using the Noise Protocol Framework)

The Bourne Shell Identity

agency and identity are confusing

- when we are logged into a shell, we consider that “we” execute the commands not the computer
- the closer a computer program is to direct manipulation, and trust through familiarity, the more we think we are “logged in as me”
- is it still running as me if I am not logged in?
- in reality “justin” on lots of computers is not me!
- it is just some computer programs we trust to varying degrees
- capabilities is a model that says “delegate to programs” and “allow them to delegate too”



there are no people in the machine

- a lot of our identity models have come from the browser and the web
- these have much more direct agency (but CSRF!)
- we have internalised this model because it is better developed
- it does not usefully apply to microservices or serverless
- think about machines not people
- “service accounts” are ways to make applications more like people, so you can add them to Active Directory

summary

summary

- capabilities are transferable authorization to something
- with more than two parties traditional ACLs are problematic
- if you have to read the code to work out if authorization is correct, you already lost
- we need to change how we do things, but this will be difficult
- just because your paper wasn’t accepted into a conference doesn’t mean it is not an important paper

questions?

@justincormack