TU Wien Informatics

Object Capabilities and Their Benefits for Web Application Security

Michael Koppmann
Software Engineering & Internet Computing

TU Wien Informatics
Institute of Information Systems Engineering
Information and Software Engineering Group
Supervisor: Privatdoz. Mag.rer.soc.oec. Dipl.-Ing. Dr.techn. Edgar R. Weippl
Assistance: Univ.Lektor Dipl.-Ing. Dr.techn. Georg Merzdovnik BSc

Problem and Motivation

- Exploiting security vulnerabilities for criminal activities has become a business which costs companies worldwide multiple billion U.S. dollars a year
- The OWASP “Top Ten” document lists the ten most common web vulnerabilities
- Wrong authorization models seem like one of the root causes
- Most applications use authorization based on Access Control Lists

User	/etc/passwd	/home/alice/secret.txt	/home/bob/shared.txt
Alice	(read)	(read, write)	(read)
Bob	(read)	()	(read, write)
Carol	(read)	()	()

Table 1: Access Control Matrix example

Methodology

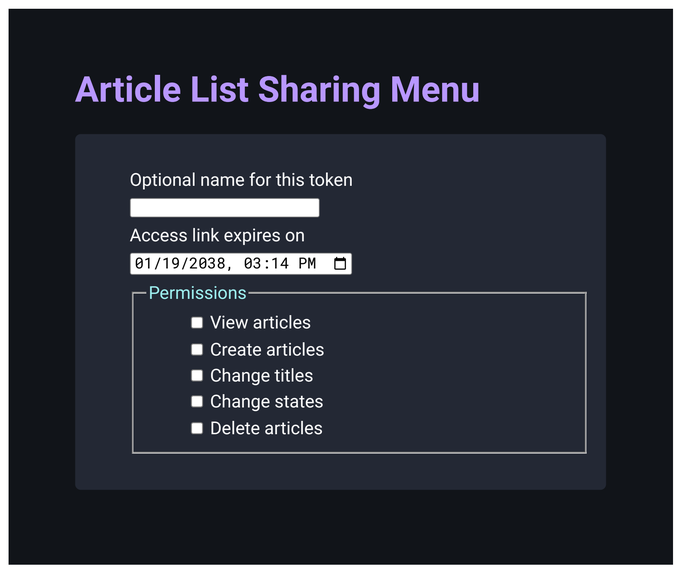
1. Design and implementation of a prototype web application based on capability-based techniques
2. Execution of a security analysis by conducting a penetration test on the prototype, based on the OWASP Top 10
3. Evaluation of the object capability model by comparing conceptual differences between OCAP and ACL
4. Evaluation of the security model by comparing differences between the prototype and existing web applications

Research Questions

- Can vulnerabilities in authorization systems be prevented by design?
- Is a capability-based system at least as secure as an ACL one?
- Can the web be used as a platform for exchanging secure tokens?
- How compatible is it with the rest of the ACL-based ecosystem?

Eselsohr – A Case Study

- Shareable access to pages with limited permissions, without requiring user accounts, with URLs as access tokens:



`https://eselsohr.example.org/articles/shared-links?acc=QMANQJKQLGW2T56NXBBHXIM5UPI5VL3GLDMCKUHTXBKKIE57I47KK7`

Comparison

Vulnerability class	Protection level: Eselsohr	Protection level: Pure OCAP system
A1:2017-Injection	◐	◐
A2:2017-Broken Authentication	◐	●
A3:2017-Sensitive Data Exposure	◐	●
A4:2017-XML External Entities (XXE)	-	◐
A5:2017-Broken Access Control	●	●
A6:2017-Security Misconfiguration	○	○
A7:2017-Cross-Site Scripting (XSS)	●	●
A8:2017-Insecure Deserialization	◐	●
A9:2017-Using Components with Known Vulnerabilities	◐	●
A10:2017-Insufficient Logging & Monitoring	○	○

●=prevention ◐=mitigation; ○=no effect; -=not analyzed;

Table 2: Security analysis results

Transfer method	Extraction method: Visible in location bar	Accessible by JavaScript	Send by default in `referer` header	Stored by default in server logs	Stored by default in browser cache
HTTP header	-	◐	-	○	○
HTTP body	-	●	-	○	●
Cookie	-	◐	-	○	●
URL path	●	●	●	●	●
URL query string	●	●	●	●	●
URL fragment	●	●	○	-	●

●=completely exposed; ◐=partly exposed; ○=not exposed; -=not applicable

Table 3: Comparison of how different data transfer methods expose web-keys

Evaluation

- Common ACL systems rely on Ambient Authority. This leads to Confused Deputy attacks which object capabilities are not susceptible to
- It is simpler to apply the Principle of Least Authority with OCAP
- Capability-based web applications mitigate common vulnerabilities
- Current browsers can either transmit data in secret but not shareable or shareable but not secret

Conclusion

- Programming with an object capability-based style can prevent certain vulnerability classes
- OCAP-based applications have no significant drawbacks compared to ACL-based applications while providing improvements in areas like shareability and embeddability
- Current browsers can be used for exchanging capabilities, but further extensions would improve their security (e.g. new URI schemes)
- An object capability application can be built with common web technologies without the need for special libraries