Client Authentication in a Federation Using a Security Mode

Sebastian Gajek and Ahmad-Reza Sadeghi
Horst-Görtz Institute for IT-Security
Ruhr-University Bochum, Germany
Problem
Terms

Phishing: stealing credentials or any other valuable information by \textit{actively or passively} creating a fake environment to deceive victims
- mounting attacks (e.g., DNS-based)
- spoofing attacks (e.g., Web Spoofing, Picture-in-Picture)

Malware Phishing: \textit{additionally} compromises the local system
- Host-file alternation
- Keylogger
- Trojan Horses
What is a Security Mode?

Security Mode
- Tame browser to ideal behavior

Predication
- Web Browsers are full of (potentially) malicious features
- Verifying security not manageable by ordinary user

Objectives
- Reduce Tampering
- User-Transparency
- Proving Security Requirements
**What is a Security Mode?**

**Security Mode**
- Tame browser to ideal behavior

**Predication**
- Web Browsers are full of (potentially) malicious features
- Verifying security not manageable by ordinary user

**Objectives**
- Reduce Tampering
- User-Transparency
- Proving Security Requirements
Case Study: Tampering

original

fake
Case Study: Tampering

![Original Image]

![Fake Image]

original = fake

W3C Workshop on Transparency and Usability of Web Authentication
Case Study: Tampering

original

fake
What is Security Mode?

Security Mode
• Tame browser to ideal behavior

Predication
• Web Browsers are full of (potentially) malicious features
• Verifying security not manageable by ordinary user

Objectives
• Reduce Tampering
• User-Transparency
• Proving Security Requirements
Case Study: Transparency

Does the right dialog really indicate a high-security failure?

SSL alert
What is a Security Mode?

Security Mode
• Tame browser to ideal behavior

Predication
• Web Browsers are full of (potentially) malicious features
• Verifying security not manageable by ordinary user

Objectives
• Reduce Tampering
• User-Transparency
• Proving Security Requirements
De-facto-Standard in Web authentication:

- Unilateral SSL authentication with username/password
- Related Work (e.g. [WS96,MSS98]) analyzed SSL in 2-Party-Scenario
- 3-Party-Protocol
  - User U (security-unaware)
  - Browser B (protocol-unaware)
  - Server S (crypto-expert)
- Recent Work [GPS05a] attempts to formally analyze browser-based protocols based on “ideal/real world” paradigm (here by [PSW00])
  - Principals are finite state machines
  - Behavior idealized
- Proved WS-Federation Passive Requestor Profile [GPS05b]
Proving Security Requirements in Browser Model

Problem
- Idealized model very complex
- Real model contains many features, not regarded

Conclusion
- If browser can be tamed to ideal behavior then one better reason about security properties
- If sender knows Browser behaves ideally and the authentication protocol is sound then sender can imply that user has been correctly authenticated

ideal browser $\rightarrow$ security mode
Candidate Solution I: Secure Mode Browser

Security Mode

- Limit the functionality → “zero-footprint”
  - User should always be aware of “what he sees is what he gets”
  - Does not solve completely phishing problem
  - Domesticates the “tools” of illusion attacks
- Non-cryptographic presentation of SSL
  - Laymen should understand SSL (e.g., [Trustbar], [Petname])
- Context-sensitive presentation of security indicators
  - Clearly highlight security alerts
  - Reduce amount of failure alerts
  - Deploy empirical results of, e.g., [XB05]
- Highlight the trustworthiness of certificate authorities
  - Today more than 70 root certificates are installed in a standard browser
  - Equally treated, but issue polices different
Example of “Online-Banking Browser”

- CA trust indicator (e.g., [Petname])
- Personalized dialogs (e.g., [DT05])
- Textual summary
- Visual summary (e.g., [Trustbar])
- Predefined links
- Context-sensitive failure information (e.g., [XB05])
Candidate Solution II: PERSEUS

Security Architecture against Malware Phishing

- Software-based security kernel (secure operating system)
- Trusted Computing (TC) functionalities
  - More and more vendors integrate a Trusted Platform Module (TPM)
- Provides elementary security properties (e.g., trusted channels, process isolation)
- PERSEUS: A generic security architecture
Candidate Solution II: PERSEUS

• Hypervisor Layer
  • Abstraction of underlying hardware (e.g., CPU, interrupts)
  • Offer an appropriate management interface
  • Enforce resource-based access control policy

• Trusted Software Layer
  • Trusted GUI secure path to applications (identify applications and thus protects against Trojan horse attacks like faked dialogs)
  • Application Manager enforces a security policy defining the applications that are allowed to be executed, measures the application’s integrity
  • Trust Manager creates and certifies keys bounded to applications
  • Storage Manager enables other applications to persistent store their states and data
Candidate Solution I+II

PERSEUS instantiations can be used to run (para-) virtualized legacy operating system (currently Linux)

Malware Process

run web browser in an isolated environment (compartment)!!!
Summary

Proof-of-Concept for Online-Banking on-going

Challenges we face

• User-friendly presentation of a trusted compartment
• Policies how to automatically activate a new compartment
• Secure and efficient migration of compartments

For more information see www.prosec.rub.de
Thank you!

Candidate Solution I
Sebastian Gajek
sebastian.gajek@nds.rub.de

Candidate Solution II
Ahmad-Reza Sadeghi
sadeghi@crypto.rub.de
References


[Petname] Ka-Ping Yee: Designing and Evaluating a Petname Anti-Phishing Tool. 2005


