Secure Internet Banking on Insecure Hosts

Christos Tziortzios

System and Network Engineering
University of Amsterdam
Agenda

- Introduction
- Research Question
- Man – in – the – Browser attack
- Solution Proposed: One – time Java Applet
- Attack Scenario
- Conclusion
- Questions

19 slides
Cybercriminals earn £48 million in 'Operation High Roller' bank hack

International fraud ring said to use automated techniques to steal €60 million from banks and commercial accounts, according to McAfee and Guardian Analytics report

26 June, 2012 09:57

A new wave of automated hacking online bank accounts might have stolen $78 million in the past year. Customers in Europe, Latin America and the United States.

This is according to researchers who poked into the computers of the hacking gang.

Bank account hacking software gets smarter

Reuters June 18, 2012 0 Comments Tweet 19 Like 28
Introduction: Cat and Mouse Game

- Evolution of attacks
  - Keyloggers
  - Man-in-the-Middle
  - Man-in-the-Browser (MitB)

- Countermeasures
  - Transaction Authentication Codes
  - 2 – factor authentication
Security vs ...

- Usability
- Marketing
- Transaction Cost

**e.g. e.dentifier2 Connected – Mode**

- Secure device
- See What You Sign
- Users may not find it usable
- Need privileges to install software
- Need for USB port
- What about internet cafés?
Is using one-time Java Applets for Internet Banking transactions a secure and usable solution?

- What kind of functionality should exist in such an applet?
- Which are the risks, related to implementing and using the previously mentioned scheme?
- Which are the strengths and weaknesses of the scheme from a security and usability perspective?
Man-in-the-Browser attack

- Malware on customer’s computer
- Real-time content manipulation

Diagram:

- 100 euro to 987654
- 200 euro to 12345
- 200 euro to 12345
- Challenge is 5678

(1)
Content Manipulation attack
Automated
Two stages
  - Manipulate data input
  - Manipulate transaction receipt
The user will never notice
Not a Man-in-the-Middle attack
  - Nothing “wrong” with the network; bar is green!
One Time Passwords, Client Certificates etc. cannot help against the attack
Man-in-the-browser attack

Points of attack
- API hooking
- Browser Helper Objects (Explorer) - Extensions (Mozilla)
- Java Script injection
- Uses regular expressions to find which content needs to be altered

Example malware
- Zeus
- Spy Eye
One – Time Java Applet

Pros

- No API hooking
- Java Virtual Machine
- No need for administrative privileges or USB
- Concepts like randomization against pattern matching
- Encryption within the applet
- Easy to push updates

Cons

- Changes what customers are used to
- Need for Java Runtime Environment; not always installed
- Transactions probably take longer (compile, sign)
- Not necessarily an answer to Man-in-the-Middle attacks
- Schemes based only on software cannot be 100% secure
What should the applet do?

- What do we need to protect?
  - Login process?
  - Transaction Details?
  - Challenge?
  - Response?

- In a compromised host all the attacker needs is the one-time codes
Possible threats: What can Malware do?

- Keyloggers
- Screenshots
- Rootkits
- Manipulate Input
- Manipulate Memory Entries
- Break a CAPTCHA
- Insert root – certificates to OS; code appears to be legitimate
- Break into Java VM
- Break Java security?
- Update botnets!
What do we want to achieve?

- Make it as hard as possible
  - 100% secure is impossible
- Prevent automation of attack
  - Make input of fraudulent data harder to automate
  - Make receipt manipulation harder to automate
Secure the applet

- Signed code
- SSL/TLS communication
  - Automatically check server fingerprint
- Secure on a lower level
  - Strings to Characters
  - Code Obfuscation: Harder to analyze code
- Graphical keyboards
- Randomize applet features
- Quick server side updates
Attack Scenarios (1)

- Attacker builds overlay applet on victim host
  - Attacker tricks the customer into using bogus applet
  - Attacker uses legitimate applet in the background
- All the attacker needs to do is make the user answer the challenge for the attacker’s transaction
  - Extract challenge from legitimate applet
  - Pass it to the customer applet
  - Let the customer generate the response
  - Use it as input for his transaction
**Attack Scenarios (2): Countermeasures**

- Sign Code and Hope(!) Java Security does not break
- Hope(!) customers pay attention to Certificates
- Randomize code
  - Make it harder to know what messages attacker must send
- Replace Strings with characters
  - Harder to manipulate the transaction receipt
- Graphical keyboards
  - Possibly harder to automate fraudulent input
Conclusion

- Software only schemes cannot be 100% secure
  - Connected mode is secure enough; use when possible
- One – Time Applet solves the problem, at least for now
  - Easy to update
- Security through obscurity to some extend
- Different levels of security – usability; functionality depends on that
- Usability Survey needed
- Penetration testing needed
Acknowledgements

- Sander Vos
- Steven Raspe
- Han Sahin
Questions

Christos.Tziortzios @ os3.nl

c.Tziortzios @ gmail.com