

# SWAP framework

## Part IV: risk management

*Kjell Jørgen Hole*



Last updated June 10, 2008

# Outline

1. Glossary
2. Threat identification
  - threat agents: crackers and disaffected employees
3. Vulnerability identification
  - attacks exploiting vulnerabilities
4. Risk evaluation
5. Risk treatment
6. Conclusions and final remarks

2

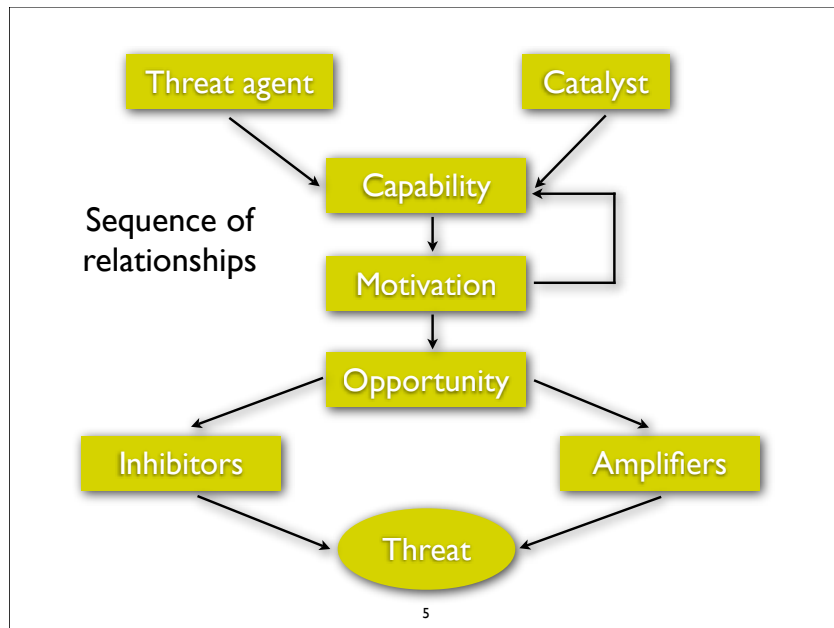
# Glossary

- **Vulnerability**—security-relevant flaw in a system
- **Attack**—a means of exploiting a vulnerability
- **Threat**—possible danger that may exploit a vulnerability
- **Threat agent**—motivated, capable individual or organization that may attack vulnerabilities in a system
- **Countermeasure**—security mechanism or procedure designed to counter one or more types of attack

3

# Threat identification

4



## Malicious hackers

- **Claim.** Malicious hackers (or crackers) are the most dangerous **external** threat agents to apps based on the framework
- crackers may work independently or for a competing commercial group
- Possible motivations for carrying out attacks
  - financial gain
  - political gain
  - revenge
  - curiosity

6

## Cracker assessment (1)

- The attack **capability** increases when individual crackers form groups with access to more information, knowledge and resources
  - the existence of such groups is well documented
- The **opportunity** to carry out an attack is considered large because applications utilizing the framework will be accessible on the Internet
- Several **inhibitors** such as fear of being caught or the perception that the application is well protected reduce the likelihood of an attack

7

## Cracker assessment (2)

- **Amplifiers** increase the likelihood of an attack at a particular time against a certain application utilizing the framework
  - period when known vulnerability exists
  - business rivalry
  - perception that an attack will not be detected
  - de-skilling through scripting

8

## Cracker assessment (3)

- A **catalyst** makes an attacker select a particular target and the time at which to initiate an attack
- A competitor, pressure group, terrorist group, or organized crime can act as a catalyst
  - these groups can use their own crackers or hire crackers from the outside
- Increasingly often, the **motivation** for an attack is often personal financial gain

9

## Disaffected employee (1)

- **Claim.** The disaffected employee is the most serious **internal** threat agent to applications based on the framework
  - manipulated by outsiders?
- Can be one of a company's hardest workers
  - shows interest in what coworkers are doing
  - volunteers for extra duties and assignments
  - work late hours and rarely takes vacations

10

## Disaffected employee (2)

- A disgruntled employee may
  - become a very capable attacker because he has intimate knowledge of the system
  - have many opportunities to attack a system because he is very difficult to spot
- As an example, an employee may try to steal a CA's private key or use an RA to create bogus certificates

11

## Vulnerability identification

12

# Possible attacks

- We discuss possible (external and internal) attacks to identify and assess potential vulnerabilities:
  1. Stealing private keys
  2. Stealing smart phones
  3. Attacking RAs and CAs
  4. Distributed Denial-of-Service (DDoS) attacks
  5. Software attacks

13

# I. Stealing private keys

- Most attackers will not waste time trying to break today's strong cryptography
- Instead, they will try to steal the cryptographic keys from where they are stored
- An individual bounded to a public-key loses his on-line identity if the private key is stolen
  - possible for the thief to operate just like the legitimate owner of the key

14

# Stealing private keys ...

- If a thief obtains a CA's private key, then he can sign bogus certificates without being detected
- A thief obtaining an RA's private key can sign bogus certificate request to a CA

15

# Countermeasure

- All private keys must be stored in hardware security modules (also called cryptographic modules)
- The private keys on the mobile phones are particularly vulnerable
  - tamper-resistant smart cards must be used to obtain an acceptable level of security
  - skilled hackers with enough resources may still be able to read information inside smart cards

16

## 2. Loosing smart phones

- People constantly loose their phones and many phones are stolen
- An attacker with access to a smart phone can
  - study all aspects of the smart phone's software and tamper-resistant smart card
  - try to brute force the PIN or password of the smart card
  - try hardware attacks on the smart card

17

## Countermeasure

- It is important that all users utilize long PINs or good passwords for the smart cards
- Must use smart cards that “close down” after receiving multiple wrong PINs or passwords

18

## Locally stored data on phone

- Remember that the secret key is determined from the PIN or password belonging to the smart card
- Try to determine password used to encrypt data:
  - dictionary attack
  - brute force attack\*
  - social engineering attack
  - shoulder surfing

\*If the attacker has access to the encrypted file and the secret key is based on a short PIN, then it easy to brute force the key

## 3. Attacking RAs and CAs

- **External attack:** attempt to steal private key using computers outside protected environment containing RA and/or CA
- **Internal attack:** employees responsible for maintaining RA and/or CA attempt to steal private key
  - greater threat than external attack (?)

20

## Protecting the root key

- Framework uses hardware security module rather than software on a CA computer to generate and store key
- National Security Agency (NSA) has developed a set of criteria
- Federal Information Processing Standard (FIPS) 140-2—*Security Requirements for Cryptographic Modules*, May 2001
- defines four levels of increasing security, use level 3 or 4

21

## FIPS 140-2 level 3

- Functional security objectives of module:
  - protection against unauthorized use
  - protection against unauthorized disclosure of non-public contents (including plaintext and keys)
  - prevent modifications
  - ensure proper operation
  - detect errors

22

## Countermeasure

- If one tries to open a module, the module destroys all cryptographic secrets in such a way that the secrets cannot be recovered
- Any attempt to open or modify the device will immediately be evident
- CA's cryptographic keys must be generated and stored in a hardware security module

23

## 4. DDoS attacks

- DDoS attacks represent a serious threat against applications utilizing the framework
- The framework architecture alone cannot defend successfully against all types of DDoS attacks
- **Countermeasure.** make sure the framework API does not simplify DDoS attacks at the application level

24

## 5. Software attacks (1)

- **Copy attack**
  - easy to download source (or binary) code because it will be freely available on the Internet
  - attacker will look for vulnerabilities such as hidden passwords, cryptographically weak random number generators, buffer overflow, etc.
- **Example:** Cracker “borrows” a phone for a few minutes, downloads modified client, and replaces old client

25

## Software attacks (2)

- **Modification attack**
  - introduce “Trojan horse” on client device
  - program tries to exploit smart card
  - program eavesdrops on client
  - captures and returns intermediate values to attacker
  - tries to change cryptographic algorithms (e.g. a key never changes)

26

## Comments

- It follows from the undecidable “halting problem” that there never will be a general test to decide whether a piece of code contains malicious code
  - only possible to detect known code by scanning for known patterns
- A high degree of non-repudiation is not possible as long as a Trojan horse can access the smart card
  - a modified compiler can introduce a Trojan

27

## Software attacks (3)

- It is quite common for applications to **leak information**
- **API attack** to obtain “internal” information
  - attack on framework API
  - use valid method calls, but in an unexpected sequence
  - utilize “unexpected” parameter values

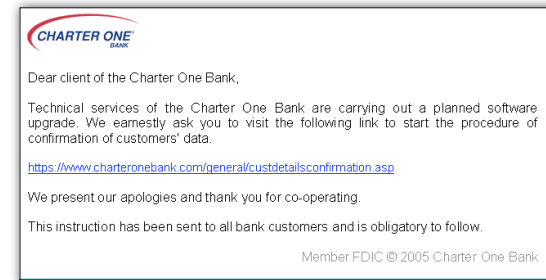
28

## Software attacks (4)

- An application can be **reverse engineered** by applying a disassembler or a decompiler to the binary code
- The disassembler produces assembly code
- The decompiler produces code in a high-level language such as C or Java
- **Java byte code is particularly easy to reverse engineer**

29

## Phishing



This phishing attempt, disguised as an official email from Charter One Bank, attempts to trick users into giving away their account information by "confirming" it at the phisher's linked website. [Wikipedia]

- Phishing e-mails entice unsuspecting users to fake web sites where they enter personal information
- About 3–5% response rate

30

## Seeding of PRNGs

- Creating truly random numbers on a deterministic device, e.g. a smart phone, is not possible
- A Pseudo-Random Number Generator (PRNG) is used to generate key material
- **Seeding the PRNG with time is a common mistake**
- Strong sources of randomness such as thermal noise and radioactive decay are needed

31

## Risk evaluation

$Risk = threat \times vulnerability \times impact$

32

## Risk in real systems

- An objective (impartial) process for assessing risk requires a "closed" system with clearly defined boundaries
- Unfortunately, real systems do not have well-defined boundaries because they interface with an ever changing world of technologies and humans
- Hence, risk evaluation is subjective, i.e., it is based on or influenced by personal feelings or opinions

33

## Dominant risks in SWAP

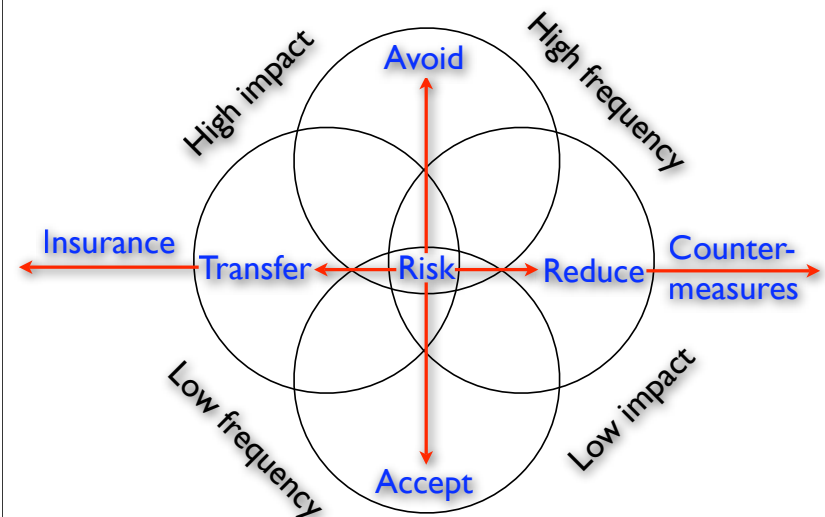
- Risks ranked in order of seriousness:
  1. Malicious client software, e.g., Trojan horses
  2. DDoS attacks
  3. Misuse of certificates
  4. Loss of keys
- These risks can be reduced, but not eliminated

34

## Risk treatment

35

### General risk treatment



36

# I. Trojan horses

1. Skilled crackers will develop more and more Trojan horses for smart phones over time
2. A Trojan may
  - generate billing events (calls and messaging)
  - steal sensitive information stored on the device
  - access smart card and sign bogus messages
3. The seriousness of the threat depends on the security model of the underlying OS
4. Framework can't thwart all attacks from Trojans

37

# Example: replacing root certificate

- Malicious software may replace the CA's root certificate on a phone's smart card
- In this case an attacker can issue his own certificates that will be accepted by the client on the phone
  - assuming client-to-client communication

38

# Malicious client software

- It is reasonable to assume that a cracker can determine the application protocol used for communication between a client and a server
- The cracker may then develop his own malicious client
- **Server must verify all inputs from the clients**
  - use a "white list" to specify allowed input

39

# What to do—signed client applications

- OS platforms move towards locking down their installation processes for third-party applications, so only *signed* applications can be installed
- some platforms may install unsigned applications after the user has received one or more warnings
- **Risk can be reduced to an acceptable level by educating customers about the threat from Trojans**

40

## 2. DDoS attacks

1. The number of DDoS attacks from large zombie networks controlled by crackers is increasing
2. All server-based applications are potential victims
  - wireless applications are especially vulnerable to attacks at the physical level
3. A DDoS attack can make it (nearly) impossible for legitimate user to access a service
4. Can't avoid all negative consequences of DDoS attacks, but it is possible to limit the damage

41

## 3. Misuse of certificates

- An ID certificate associates a public key with an identifier pointing to the keyholder
- The signed certificate does not prove that the owner of the certificate is who he claims to be
  - it provides only a level of confidence in the claim
- There is a risk that certificates will be misused
- This risk is much smaller than the risk associated with the traditional use of passwords or PINs

42

## 4. Loosing keys

- The risk of loosing private keys can be reduced to an acceptable level by utilizing:
  - smart card storage on phones
  - CA with hardware security module placed in a secure computer room
  - application server with smart card storage placed in a secure computer room

43

## Conclusion

- The likelihood of serious attacks depends on the number and types of applications using the framework
- Assuming that the framework becomes popular, one must take for granted that the framework will experience serious attacks from crackers
- Information on how to carry out an attack is likely to spread on the Internet, resulting in attacks on other applications based on the framework

44

## Final remarks

- We have not considered the physical security needed to protect RAs and CAs against hardware errors, loss of power, flooding, vandalism, and terrorist attacks
- This security analysis must be repeated on a regular basis because vulnerabilities and threats change over time

45

## Sources

- A. Jones and D. Ashenden, *Risk Management for Computer Security*, Elsevier, 2005
- J. R. Vacca, editor, *Public Key Infrastructure: Building Trusted Applications and Web Services*, Auerbach Publications, 2004
- J. Viega and G. McGraw, *Building Secure Software*, Addison Wesley, 2002
- I. Winkler, *Spies Among Us*, Wiley, 2005
- National Research Council, *Who Goes There?* The National Academic Press, 2003

46



47