

SWAP framework

Part III: framework analysis

Kjell Jørgen Hole



Last updated June 10, 2008

Outline

1. Can we use “naked keys” instead of certificates?
2. When can revocation be simplified?
3. Logging of framework incidences
4. Suspension of certificates?
5. Data sanitization
6. Privacy issues

2

Alternative architecture inspired by SSH

3

Alternative architecture

1. We'll first argue that public-key certificates are needed to support
 - a simple introduction of new servers as the number of clients grows
 - client-to-client communication
2. Then, we'll assume a pure client-server model with a limited number of users and discuss a simpler alternative architecture inspired by the SSH protocol

4

“Naked keys” reduces scalability

- Assume that the framework does without certificates and operates only with “naked keys”
- If a root CA is not used, then it is more complicated for a client to get the public key belonging to a new application server
 - the client may have to revisit an RA
- In other words, it is more difficult to serve an increasing number of users in an efficient manner

5

Direct client-to-client communication

- If two clients exchange “naked keys” then it is possible for a third party to substitute its own keys
- Must use certificates signed by a trusted CA to defend against man-in-the-middle (MIM) attacks

If direct client-to-client communication is not needed and the number of clients is relatively small, then public-key certificates are not needed

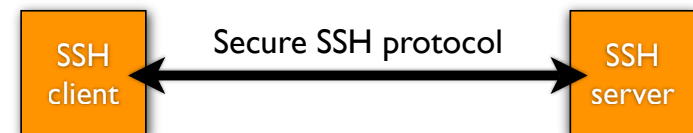
6

An SSH primer

- SSH (Secure SHell) is a protocol providing authentication, encryption, and integrity of user data
- SSH-2 is the latest version of the protocol
- SSH has a client-server architecture
 - an SSH server program is typically run by a system administrator
 - users run SSH client programs on their local computers

7

What can SSH do?



- Secure remote logins—program `ssh`
- Secure file copying—program `scp`
- Port forwarding—rerouting of a TCP/IP connection through an SSH connection

8

Technology basis

- SSH is based on public-key (or asymmetric) cryptography, just like SSL
- SSH uses “naked keys,” unlike SSL that utilizes public-key certificates
- **Problem:** since there is no trusted CA, SSH is vulnerable to MIM attacks the first time a client connects to a server

9

Simplified architecture

- Strong initial identification can remove the problem of MIM attacks
- It is important to bind the initial public key to a user and to store this information in a central database
- Old keys can then be used to distribute a new keys

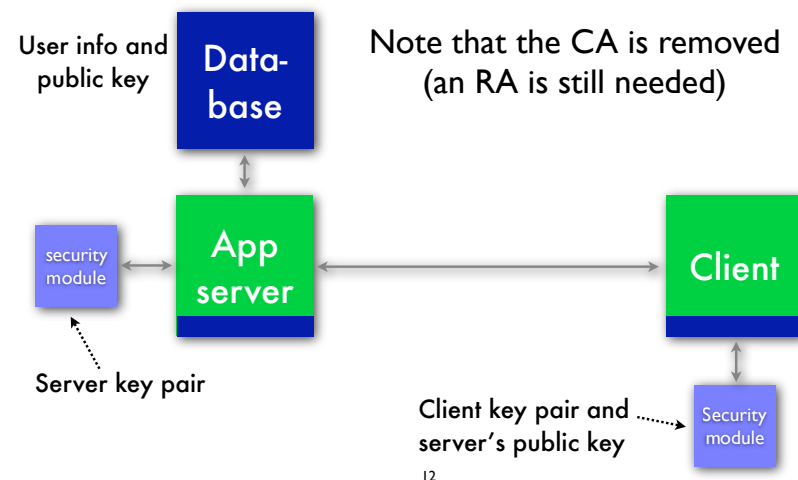
10

Observations

- Store keys in encrypted file or on a smart card
- User enters password to decrypt keys in file
 - attacker who knows the public key can brute force a weak password by trying different passwords until the correct key is returned
- must also keep public key secret
- Must store old keys for a long time because some users do not run application very often

11

Alternative architecture



12

Our choice

- We choose to use public-key certificates because we want to support
 - client-to-client communication
 - a large number of clients

13

Revocation revisited

The cost of public-key systems is dominated by the cost of key revocation.

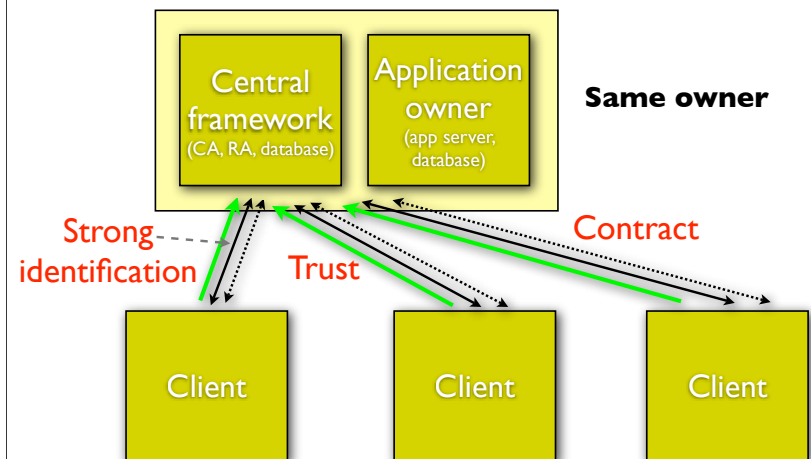
14

When can the revocation be simplified?

- We again consider a framework utilizing CAs and certificates
- In Part II we discussed the use of a revocation service providing clients and app. server with the revocation status of certificates
- We now assume that the same company owns both the CAs and the application server
 - each user must trust both the root CA and the application server

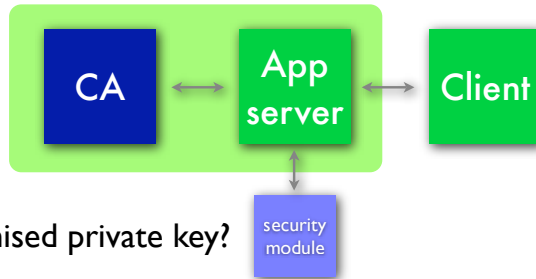
15

Simplified trust relationships



16

Same owner



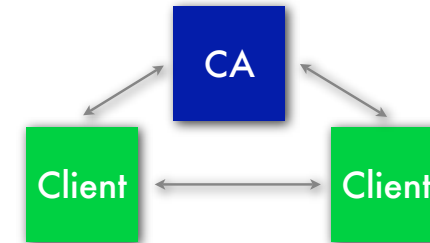
Compromised private key?

- **Simplified revocation mechanism:**

1. Client is denied access to the app server if the server's private key is compromised
 - *client does not need to verify the revocation status of the server certificate*

17

2. App server asks CA for revocation status of client certificate
3. During *client-to-client communication* each client asks CA to verify the revocation status of the certificate from another client



18

Advantages of simplified revocation

- Less overhead associated with revocation
- Less vulnerable to DDoS attack because server and CA can communicate behind a firewall

19

Robust logging

20

What to log (1)

- The framework provides **secure communication**, **secure data storage**, and **certificate management** services
 - it is not concerned with the content of the data messages
- The framework only logs framework incidences
 - the application server may log application specific incidences

21

What to log (2)

- Security-related system activity:
 - RA registration attempt
 - certificate issuance
 - certificate update
 - request for revocation information
 - certificate revocation request
 - change of revocation status

22

What to log (3)

- Security-related exceptions:
 - cryptographic operation that failed because of an attempt to use a terminated (or nonexistent) key
- Operator activities:
 - start up and shut down of OS
 - start and shut down of RA/CA software
 - change of software
 - system change made by operator

23

How to log (1)

- It seems natural that the logging is done by each on-line CA and that the results are stored in a (central) file or a database
- The log must have integrity protection
- Each entry event must have a time stamp and the IP address of the event's source
- The log must be backed up

24

How to log (2)

- Real time log
- Can inspect log at any time
- Only authorized operators have access to log

The logging (and monitoring) of incidences must support the incidence response plan

25

Incidence response (1)

- Reaction depends on the incidence and the application running on top of the SWAP framework
- Possible reactions are:
 - more detailed logging
 - blocking a single user or group of users
 - limiting or shutting down database access
 - shutting down application server or framework

26

Incidence response (2)

- It may be necessary to
 - search for malicious code on clients and servers
 - verify integrity of software and databases
 - rebuild (parts of) system
 - change keys

27

Suspension of certificates

28

Suspension of certificates

- The framework administrator should be able to *temporary* suspend certificates
- Re-activation of a suspended certificate can only be done after the user has authenticated himself
- All suspended certificates that are not reactivated within X days should be revoked

29

Why allow suspension of certificates?

- When a certificate is revoked, it is necessary to go through the complete initialization process one more time
- In some cases it may not be completely clear that a certificate should be revoked
- It is better to suspend the certificate in this case because this approach reduces the burden on the users

30

Data sanitization

31

Output filtering

- An attacker may carefully examine the output from the framework to determine the effects caused by different inputs
- Output filtering is used to limit the amount of information available to the attacker
 - prohibits verbose error messages
 - blocks unexpected output

32

Input filtering

- Systems are most frequently attacked through entry points
- Use input filtering to resist attacks
 - check input for length, data type, acceptable values
- It is particularly important to use input filtering at *external* entry points to the framework

33

Filtering between components

- The framework consists of many components
- In accordance with defense in depth, flows between components are sanitized

34

Privacy issues

35

Privacy in the context of authentication

- It is important to take the users' privacy concerns into consideration during the design process (of any authentication system)
- Individual authentication makes it possible to monitor people's use of a system
- If the same ID is used in multiple systems, then it is possible to determine people's habits

36

Authentication in SWAP

- It is necessary to use individual authentication in online banking and health care systems because accountability is needed
- Because the SWAP framework only supports a single application, it is not possible to determine users habits in multiple domains

37

Design principles (1)

1. To limit the privacy concerns, avoid the use of identifiers utilized in many systems such as SSNs
2. Collect only the minimum information needed for authentication
3. Only collect personal information with the full knowledge and consent of the individual
4. Collected information should be relevant, accurate, timely, and complete

38

Design principles (2)

5. Use of data should be specified at the time the data are collected
6. Data should only be used for the specific purpose for which they are collected
7. Any user has the right to access and correct the information about him- or herself

39

Conclusions

- The framework is based on a PKI to obtain
 - secure client-to-client communication
 - scalability
- Revocation overhead is reduced by considering a model where the framework and the application server have the same owner
- Suspension of certificates is useful
- Data sanitization is important

40

Sources

- D. J. Barrett, R. E. Silverman, and R. G. Byrnes, *SSH, The Secure Shell*. O'Reilly, 2nd edition, 2005
- BSK, Norsk BankID sertifikatpolicy for banklagrede kvalifiserte sertifikater til personkunder, v 1.1, desember 2005
- K. Kenan, *Cryptography in the Database*. Symantec Press, 2006
- National Research Council, *Who Goes There?* National Academies Press, 2003

