1.264 Lecture 29

Secure Sockets Layer (SSL)
Security summary

Next class: Exercise due after class

Case study

- Write protocol that Andrews and BBI use
  - B->A:
  - A->B:
  - B->A:
  - A->B:
- List 3 attacks that C can attempt:
  - Man in the middle
  - Spoof/impersonation
  - Brute force
  - Others

Solution

- B-> A: $K_{B,}$ A, B, T
- A-> B: $\{K_{AB}\}_{KB,}$ A, B, T
- B-> A: $\{N, M_{BA}\}_{KAB}$
- A-> B: $\{N, M_{AB}\}_{KAB}$
- Attacks:
  - Spoof: C can spoof A's IP address (identity), send different $\{K_{AB}\}_{KB}$ to B. C can then send and receive bogus messages
  - Brute force: C can break the 56 bit key
  - Man in middle:
    - C can spoof B and send bogus K_B' to A.
    - When A sends K_{AB}, C can decrypt it with private K_B.
    - C sends $\{K_{AB}\}_{KB}$ to B. B does not know K_{AB} is from C.
    - C spoofs A's address, can decrypt messages between A and B

Encryption

- Secure Sockets Layer (SSL) or its successor, Transport Layer Security (TLS), use both symmetric and asymmetric encryption
  - Asymmetric encryption is used to establish a secure channel using public-private keys between the two parties who wish to communicate
  - Certificates/public keys are exchanged
    - The parties don't have to be known to each other, but must be known to a common authority issuing certificates
  - The asymmetric channel is used to exchange a symmetric key generated by the client
  - The rest of the session uses the symmetric key, which is much faster
  - Other safeguards are used to protect against replay, etc.

Secure Sockets Layer (SSL)

- Implements core security for Internet
- Dominant protocol for browser-server communications
  - Standardized as Transport Layer Security (TLS)
  - TLS 1.2 is current version, extended from SSL 3.0
- Many choices in symmetric algorithm, message digest and authentication.
  - Uses RSA public key asymmetric algorithm.
- Client and server negotiate strongest common protocol
- SSL has built-in compression
  - Encrypted message has no patterns and can't be compressed, so compression must be done before or within SSL, or not at all
- SSL encrypts all client-server communications
  - Only public info available is that client is talking to this server
    - IP protocol must be modified to mask endpoints (IPsec)

SSL protocol steps

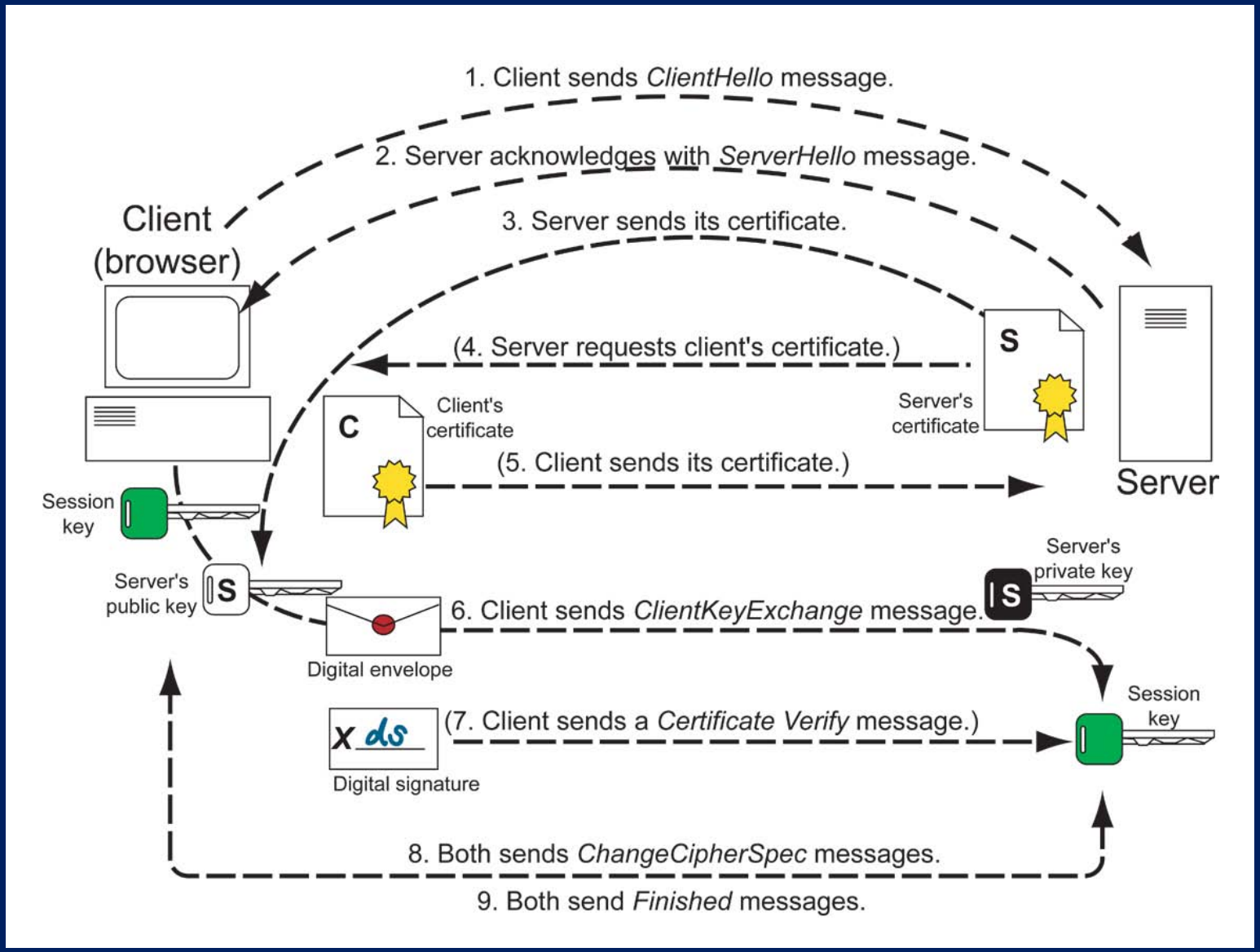


Image by MIT OpenCourseWare.

SSL protocol steps

1. ClientHello: list client capabilities: SSL version, algorithms
2. ServerHello: chosen algorithms, and nonce
3. Server sends its certificate
   - Client can verify server certificate with root CA in browser

4-5. Optional, used in business-business secure connections

6. ClientKeyExchange: Client generates trial symmetric key, encrypts it with server public key and sends it to server
   - This prevents a listener (snooper) from copying past message (spoofing)

7. CertificateVerify: Optional, used b-to-b to authenticate "client"
8. ChangeCipherSpec: Confirm session key and cipher to be used
9. Finished: Client and server message digest entire conversation to ensure all messages were received intact
10. Client and server switch to encrypted mode using symmetric session key

All of this happens before you see the XHTML page requested.

Certifying authorities and public keys

- "Obvious" ways to obtain public keys don't work
  - Can't keep all possible recipient keys on your system
  - Can't ask recipient to send it to you, because he/she might be spoofer
  - Don't (can't?) have large public database:
    - Costs, performance, security for billions of keys a concern
- Certifying authorities (CAs) are used instead
  - CAs are commercial entities, whose public keys are in your browser
  - TLS asks server to send you its digital certificate, signed by a CA, before you communicate with it
  - From certificate, TLS verifies server identity and gets its public key
    - CA encrypts server certificate and hash with CA's private key
  - TLS decrypts server certificate with CA public key and checks hash to be sure certificate has not been altered

Obtaining a digital certificate for a server

1. Generate a public/private key pair on your system
2. Keep private key, send "certificate request" to CA:
   Includes public key and identifying information about you
3. Pay CA fee
4. CA verifies your identity, cursorily or extensively
5. If you are ok, CA creates certificate body with your public key and ID info:
   Server ("site") certificate has URL
   Browser ("personal") certificate has name and email address
6. CA generates message digest from certificate and signs it with its private key, creating the actual certificate
7. CA sends certificate to you.

Root CAs and certificate chains

- Root CAs have certificates on browser or Web server
  - Must believe Verisign, etc. and your system vendor are ok
- Root CAs can sign other CA's public keys
  - Signature includes the root CA's certificate
  - Chains of certificates can be created
    - Last certificate contains certificates of every CA in the chain, so it can be traced back to the original root CA
    - You use first CA's public key to get 2nd CA's public key, which you use to get 3rd CA's public key, etc.
    - Chains typically used in intranets today to verify browsers via a chain of servers
- It's reasonable to generate and administer your own public and private keys when number of sites is limited
  - MIT manages its own keys, as do many corporations

Certificate problems

- Events invalidating public/private key pair
  - Theft, change of ID info, compromise of key
  - Disk corruption (private key is encrypted via password on disk)
  - Certificate revocation list (CRL) not checked
  - Certificates generally expire in a year, but that's a long time…
- Privacy is impossible, because you are identified to other party

Exercises

1a. Examine Web site security for 5 sites

- Bank: fidelity.com and your own bank (or hsbc.com)
  - Firefox: Right click, View Page Info, Security, Certificate
  - IE: Right click, Properties, Certificate
- Transportation carriers:
  - amtrak.com or fedex.com
  - aa.com or ups.com
- One other organization, of your choice (e.g. Google)

1b. Record for each site and browser

- Symmetric key algorithm and key length (e.g., RC4 128)
- Asymmetric key algorithm and key length (e.g., RSA 1024)
- Message digest algorithm (thumbprint) (e.g. sha1)
- http://www.digicert.com/help/

1c. Record anything unusual

- Do any pages send your password in the clear?
- What could an attacker in an Internet café do?

2a. Look at the certificates in your browser

- Firefox: Tools->Options->Advanced -> Encryption
- IE: Tools-> Internet Options -> Content

Solution

- Fidelity: Home page secure
  - Chrome: AES256, RSA2048 (1024 last year), sha1
  - IE: AES 128, RSA2048 (1024 last year), sha1
- HSBC: home page not secure
  - Many banks are now secure. Much different than in past.
- Amtrak, FedEx:
  - Sends username, password in clear
  - Internet café hacker can perform man in middle attack
- UPS:
  - Home page not secure
  - Login link to https page; sent username, password in clear in past (for many years)
- AA: home page not secure (alternates every year!)
- Google: RC4-128, RSA2048, sha1
- Certificates in browser
  - Just MIT personal certificate, typically
  - Many CA, server certificates. Some revoked, fraudulent (IE)
- Make your bookmarks point to https://xyz.com , not http://xyz.com if you're accessing a secure site

Why doesn't all of this work?

- No single trusted authority for certificates
  - Client or server side
- Certificate issuance process is fairly weak
  - Weak tie to financial or government-issued identity
- Internet is open to entire world to exploit any weakness in people, protocols (process), systems
  - People issues and compromises are significant risk
- How can we manage these risks better?
  - Use a more limited network for key transactions
    - Fewer people/principals involved, lower risk of compromise
  - Single trusted (and trustworthy) authority for the network: telecom carrier
  - Known other parties
  - Whole world can't try to break you directly
    - Some flaws will remain unexploited

Security summary: Four dimensions of security

- People
  - Motivation, skills, communication, expectations, stakeholder buy-in, …
- Protocol (Process)
  - Risk assessment, requirements, design, schedule, QA
- System (Product/Definition)
  - Selected set of software and hardware components
  - Match needs, avoid goldplating/feature creep, avoid immature technology, …
- Technology
  - Encryption, biometrics, detection, ….
  - Avoid silver bullets, avoid frequent changes, …

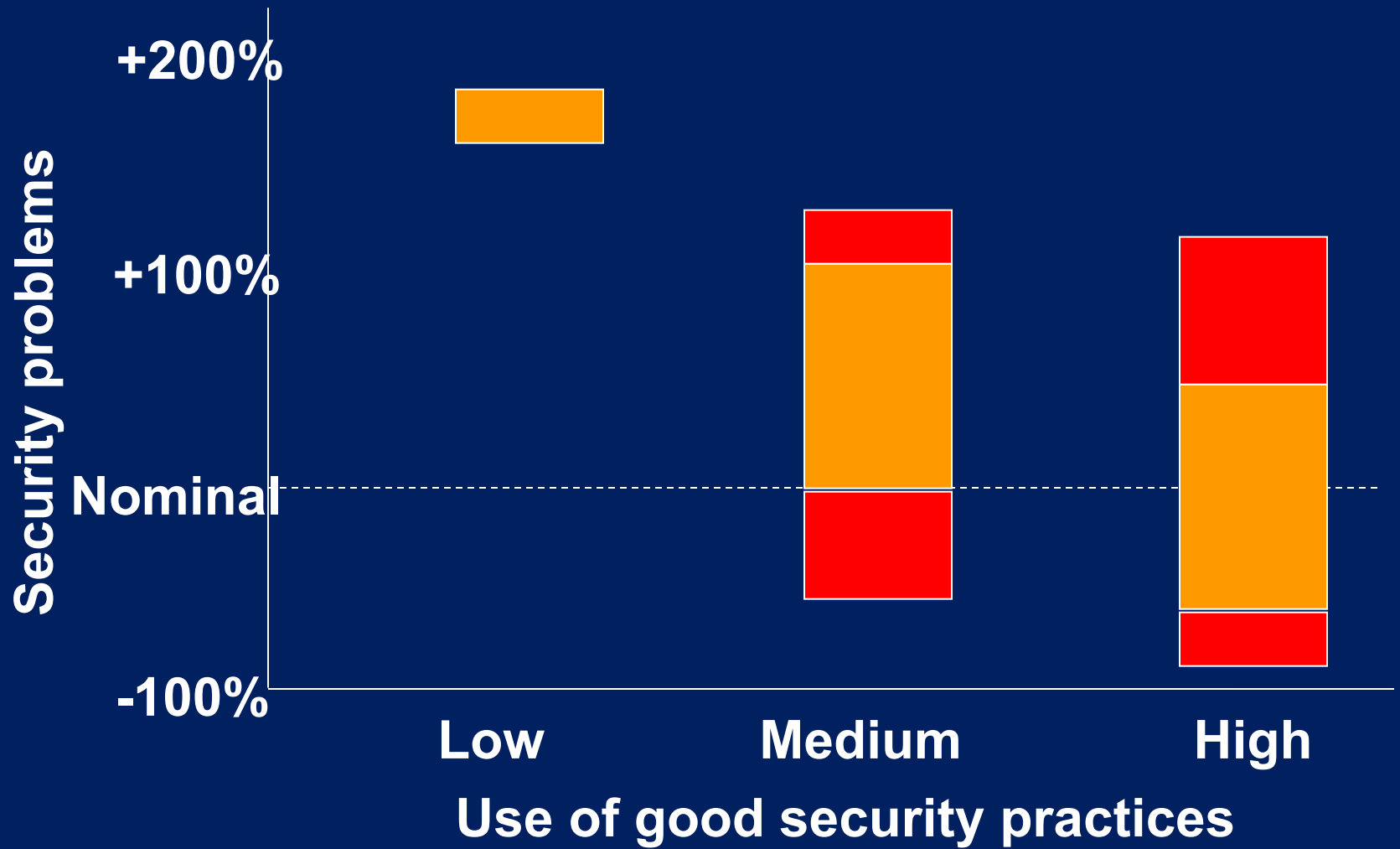
Security risks

- Essentially the same across people, process, product, technology (physical and electronic) threats
  - Identification: staleness, replay, cut and paste, ….
  - Authorization: breaking privileges, altering data, breaking physical security, …
  - Denial of function: services, monitoring, data quality, …
  - Tampering/interception: communications, shipments, items
- Many security processes span people, information and physical elements
  - Weaknesses often exist at the interfaces between them
- Protocols to handle people, process, product and technology threats are similar

Managing threats

- Tolstoy (Anna Karenina) again:
  - "Happy families are all alike; every unhappy family is unhappy in its own way."
- Secure organizations are all alike; every insecure organization is insecure in its own way
  - Doing everything capably provides security (happiness)
  - Doing everything perfectly but one thing terribly typically leads to insecurity (unhappiness)

Doing a few things right is not enough



Success (10 ok) vs failure (9 perfect)

Security process

Technical fundamentals

Spiral model as basis for development

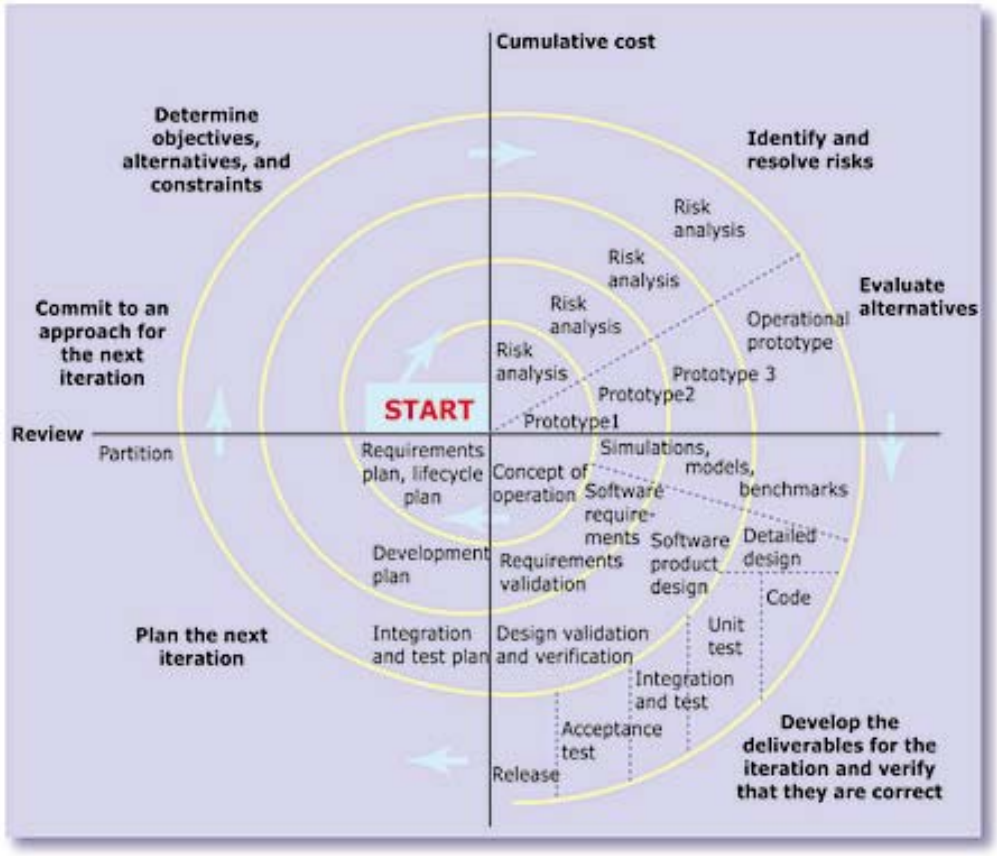


Image by MIT OpenCourseWare.

Objectives, alternatives, constraints

- Unified modeling language (UML)
  - Use cases and scenarios
  - State diagrams
  - Sequence and activity diagrams
- UML:
  - Is appropriate for modeling and understanding typical security issues with mixed computer system, physical and manual processes
  - Can document normal processes and conditions as a baseline against which we can measure and detect differences (deviations)

How to infiltrate a transit system

- Intruder enters via fire exit or maintenance access that has conventional lock. No smartcard, no camera because vulnerability is too low. He gets into tunnel with low light conditions. He then…
- Intruder enters station and hides in storage area or unused booth or... At night, when there is no lighting he sets something in place for the next day…
- Intruder waits for a stormy night. He sets off an alarm at portal. Security comes and finds nothing. He waits a half hour and does it again. Security doesn't bother. He then…
- Intruder goes to transit station and leaves smoke grenade on timer. When fire service responds, he comes in with them during the general chaos and…
- The attacker disables non-redundant communications from a sensor. Repair doesn't occur until the next day. He enters a tunnel and…
- The attacker calls claiming to be from the camera or sensor vendor and needs the serial number on a unit. It's given to him, the staffer not knowing the serial number is also the crypto key for the communications with the unit. The attacker buys or steals a similar unit, reprograms it, and attaches it to the network. The rogue device continues to report 'all is well' as…
- The attacker breaks a device or comm line, waits to see what security personnel arrive, and waits to see them leave. The device won't be fixed until the next morning. He has several hours to…

UML use case: How to infiltrate a transit system

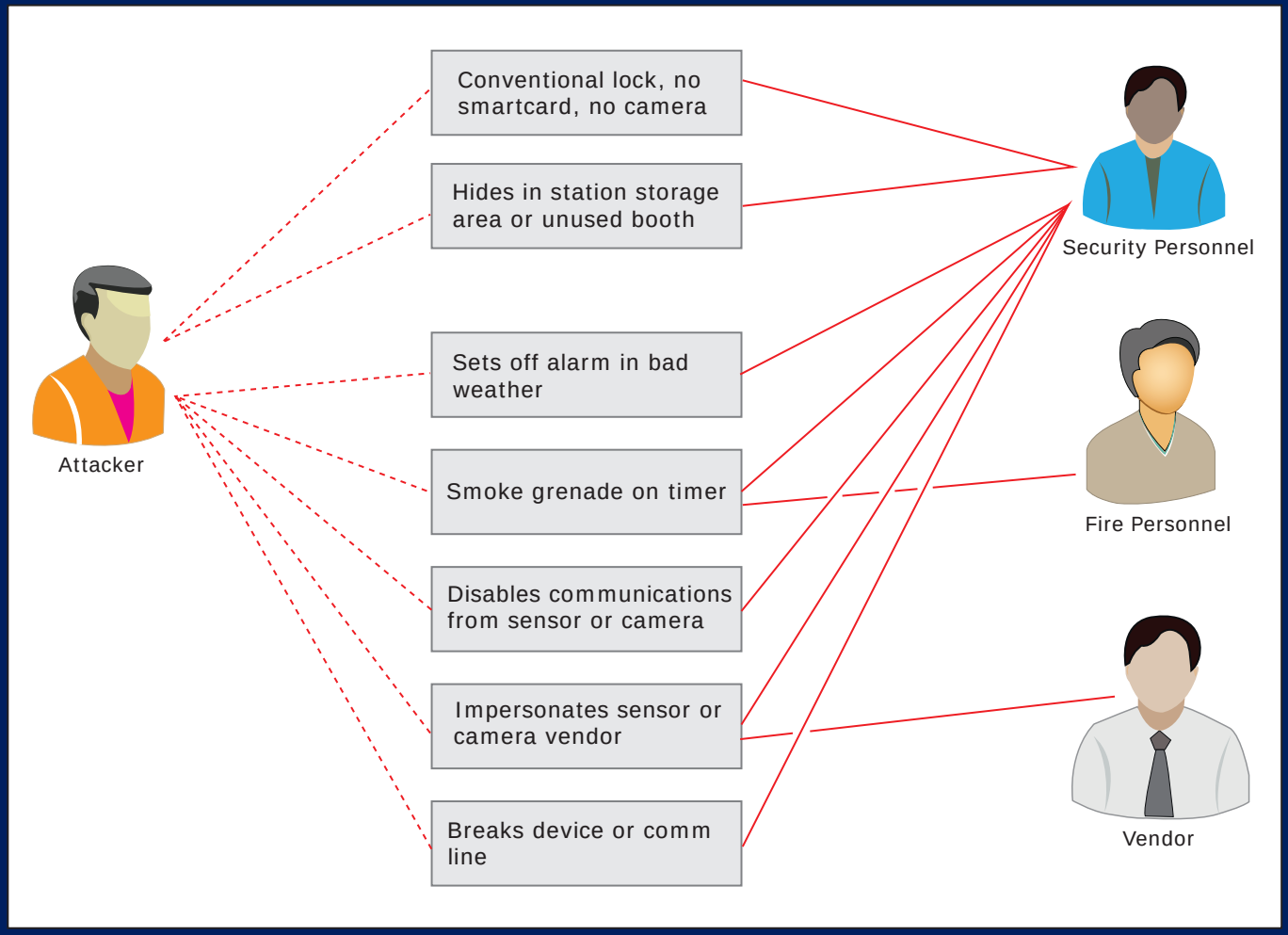


Image by MIT OpenCourseWare.

How to compromise transit access control

- Rogue employee continues to extend expired employee cards
  - Banks require all staff to take at least 10 consecutive days vacation every year, for change of control
- Rogue employee creates fictitious employee, issues card
  - Learns HR system is down once a month and there is a gap in verification
- Rogue employee takes over identity of part time employee or one on sick leave
  - Changes contact info to another employee in collusion. Attempts to break into systems; can do so anonymously
- Multi-agency access cards.
  - Procedures will not be identical, gaps will exist, and rogue employee can take advantage of them
- Excessive processing errors in issuing, reissuing or revoking cards
  - Rogue employee can acquire cards
- Mailing verification documents or cards.
  - Loss and compromise rates can be high. Rogue employee raids mailbox.

State diagram: transit access control

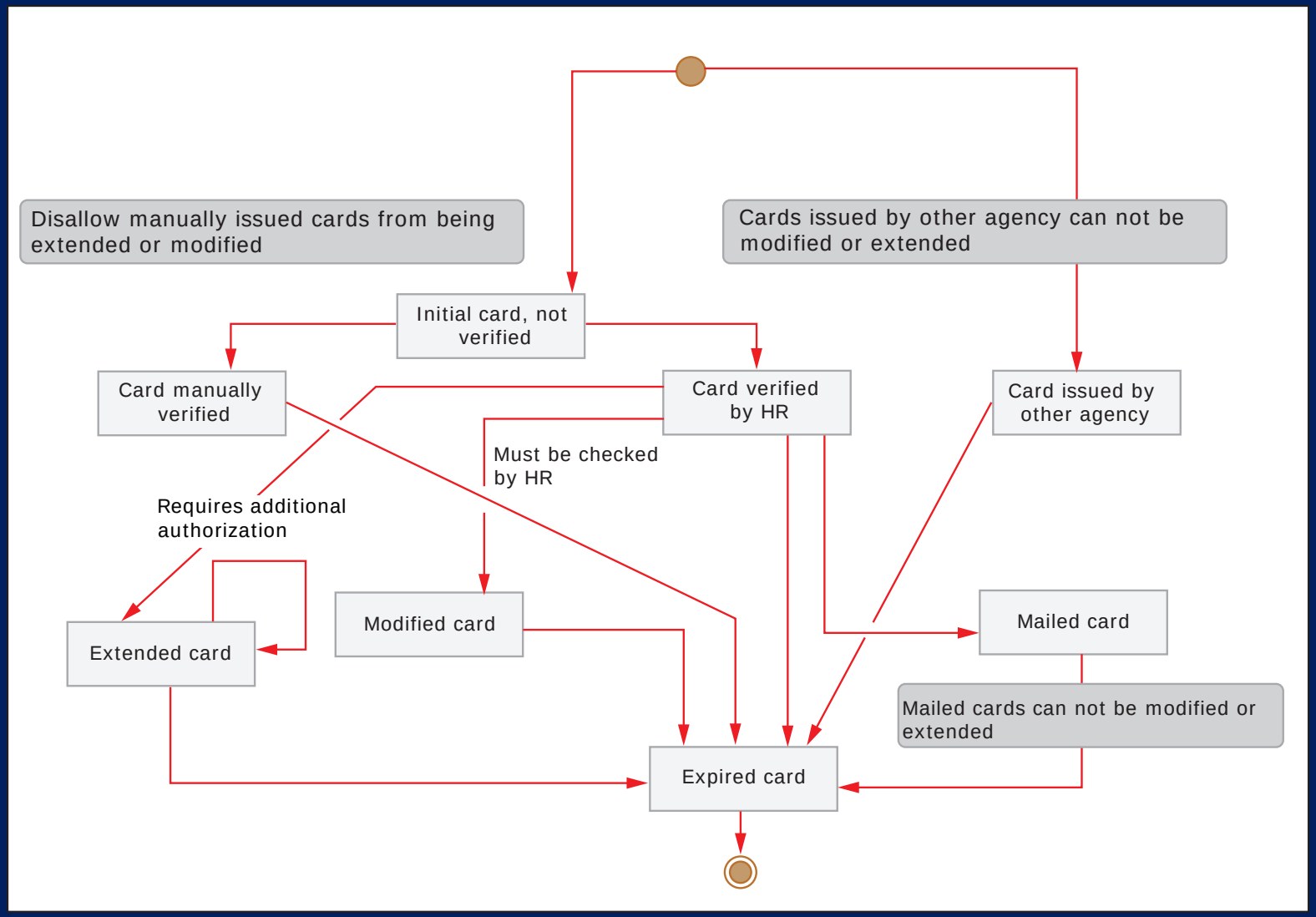


Image by MIT OpenCourseWare.

Use cases: access and other controls

- Ask these questions: Create use cases relevant to your organization based on these questions:
  - Can a single person control the data, decisions and monitoring of any process for which the person has decision rights?
    - If so, preventive controls are lacking.
  - How would you know if someone decided to disrupt your transit or IT operations?
    - A positive answer to this question identifies a detective control.
  - Can you prevent an unauthorized person from accessing this process or hiding abnormalities?
    - Do you have a way to detect unauthorized access?
    - (E.g., logging server for HIPAA)
  - Is there a system to identify all unusual events?
    - Is there a way to alert management if an unusual event is not reported or addressed?
    - Is critical information backed up?
  - Is staff evaluated for compliance with policies and procedures?

1.264J / ESD.264J Database, Internet, and Systems Integration Technologies
Fall 2013