TCB Design

Design principles

- Unbypassable (completeness): there must be no way to breach system security by bypassing the TCB.
- Tamper-resistant (security): TCB should be protected against other parts outside the TCB. These parts cannot modify the TCB's code or state.
- Verifiable (or correctness): it should be possible to verify the correctness of TCB.

Size of TCB

- A system with a smaller TCB is more trustworthy and easier to verify (we do not need to make too many assumptions, which may be violated). This follows the KISS (Keep It Simple, Stupid) principle
- Designing a secure system with a smaller TCB is more challenging (we need to consider more malicious entities)

Attacker's Assumption

Type of attacker

- Active: manipulate or disrupt the systems, e.g., modifying data, injecting code
- Passive: observing and gathering information without interfering system

Attacker's knowledge

- Know the system's design, architecture, source code, etc. ,
- Lack the detailed knowledge and must rely on probing or trial and error

Attacker's capability

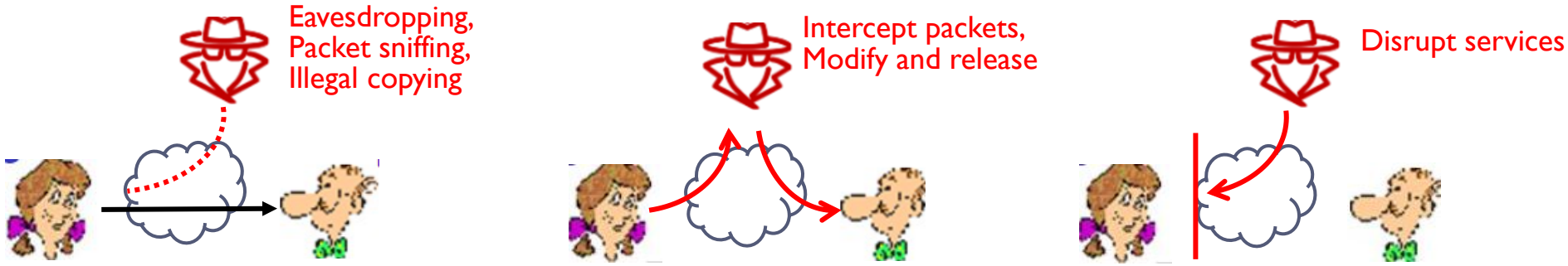
- How much computing resources can the attacker leverage?
- What parts of the system can the attacker interact with?
- Does the attacker have unlimited time or need to act quickly?

Security Properties

The security goals that we aim to achieve for the system.

Common security properties (CIA model)

- Confidentiality (C): prevent unauthorized disclosure of information. Sensitive information should not be leaked to unauthorized parties
- Integrity (I): prevent unauthorized modification of information. Critical system state and code cannot be altered by malicious parties
- Availability (A): prevent unauthorized withholding of information or resources. The resources should be always available for authorized users



Security Properties

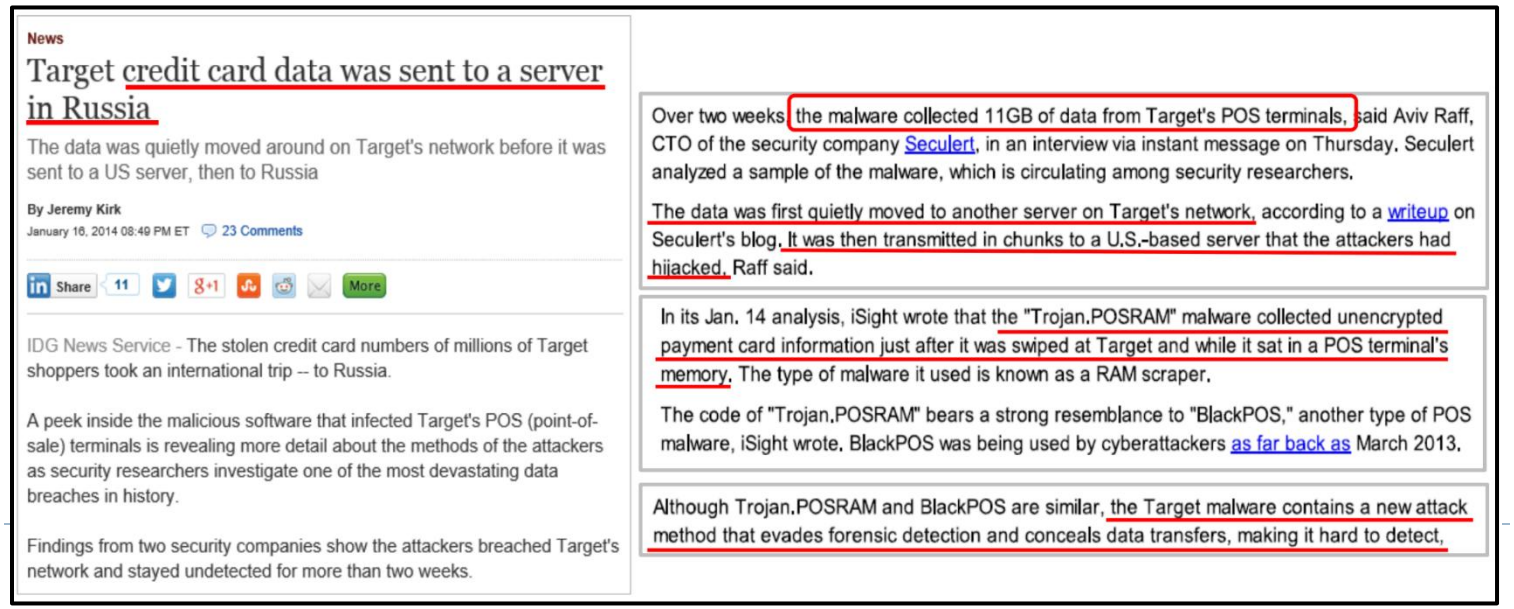
Other properties

- Accountability: actions of an entity can be traced and identified
- Non-repudiation: unforgeable evidence that specific actions occur
- Authenticity: ensure the communicated entity is the correct entity.
- Anonymity or privacy: hide personal information and identity from being leaked to external parties.
- Verifiability: the system's operations can be independently verified.
- Freshness: the data or communications are current and not reused or replayed.
- Fault tolerance: the system can continue to function correctly despite failures.

Case Study: Threat Model of Target Attack

Threat Model

- Trusted Computing Base: the Target computer system including the OS and hardware is trusted. However, the malicious software is not trusted, which leaks the data to the attacker
- Adversarial capabilities and knowledge: the attacker can launch malware on the Target's POS, and collect the credit card data stored in the database.
- Security properties: we consider the confidentiality: protecting the system from leaking sensitive information.



News

Target credit card data was sent to a server in Russia

The data was quietly moved around on Target's network before it was sent to a US server, then to Russia

By Jeremy Kirk
January 16, 2014 08:49 PM ET 23 Comments

Share 11 More

IDG News Service - The stolen credit card numbers of millions of Target shoppers took an international trip -- to Russia.

A peek inside the malicious software that infected Target's POS (point-of-sale) terminals is revealing more detail about the methods of the attackers as security researchers investigate one of the most devastating data breaches in history.

Findings from two security companies show the attackers breached Target's network and stayed undetected for more than two weeks.

Over two weeks the malware collected 11GB of data from Target's POS terminals, said Aviv Raff, CTO of the security company Seculert, in an interview via instant message on Thursday. Seculert analyzed a sample of the malware, which is circulating among security researchers.

The data was first quietly moved to another server on Target's network, according to a writeup on Seculert's blog. It was then transmitted in chunks to a U.S.-based server that the attackers had hijacked, Raff said.

In its Jan. 14 analysis, iSight wrote that the "Trojan.POSRAM" malware collected unencrypted payment card information just after it was swiped at Target and while it sat in a POS terminal's memory. The type of malware it used is known as a RAM scraper.

The code of "Trojan.POSRAM" bears a strong resemblance to "BlackPOS," another type of POS malware, iSight wrote. BlackPOS was being used by cyberattackers as far back as March 2013.

Although Trojan.POSRAM and BlackPOS are similar, the Target malware contains a new attack method that evades forensic detection and conceals data transfers, making it hard to detect,

Security Strategies

Prevention

- Take measures that prevent your system from being damaged

Detection

- Take measures so that you can detect when, how, and by whom your system has been damaged.

Reaction

- Take measures so that you can recover your system or to recover from a damage to your system.
- Always assume that bad things will happen, and therefore prepare your systems for the worst-case outcome

Design Principle: Least of Privilege

Assign privileges carefully:

- Give each entity the minimal permissions to complete the task.
- Give the privilege when needed, and revoke the privilege after use
- The less privilege that a program has, the less harm it can do if it goes awry or becomes subverted.
- If granting unnecessary permissions, a malicious entity could abuse those permissions to perform the attack.

Examples:

- Never perform personal activities using root or admin account in an OS
- A photo editing application on a smartphone is only allowed access to the gallery but not the microphone or location.

Design Principle: Separation of Privilege

Split the responsibility:

- To perform a privileged action, it require multiple parties to work together to exercise that privilege, rather than a single point of control or decision.
- Minimize the risk of misuse, error, or compromise by ensuring that no single entity has full control over critical processes

Examples:

- In a financial system, transferring large sums of money requires approval from an employee (initiator), and additional approval from a manager (reviewer).
- A developer writes code but cannot directly deploy it to production; deployment is handled by a separate operations team