
# *Insider Threats in the SDLC*

## Lessons Learned From Actual Incidents of Fraud, Theft of Sensitive Information, and IT Sabotage

**Dawn M. Cappelli**
**Randall F. Trzeciak**
**Andrew P. Moore**

# Financial Institution Discovers $691 Million in Losses…

*End User Evades Auditors for 5 Years by Modifying Source Code*

# Customers Report Strange Disruptions in Telecommunications Firm's Operations...

*Malicious Code Planted One Year Ago by Former Employee Modified Company's Communications Protocol*

# COULD THIS HAPPEN TO YOU?

# Overview of Talk

Purpose of this presentation

Evolution of CERT's insider threat research

Insider threats during the software/system development life cycle (SDLC)

Common Sense Guide – Best Practices

# *Purpose of This Presentation*

# Evolution of CERT Insider Threat Research

Insider threat case studies

- U.S. Department Of Defense Personnel Security Research Center (PERSEREC)
- CERT/U.S. Secret Service (USSS) *Insider Threat Study*

Electronic crime surveys

- *ECrime Watch* conducted with CSO Magazine and USSS

Best practices

- Carnegie Mellon CyLab *Common Sense Guide to Prevention and Detection of Insider Threats*

System dynamics modeling

- Carnegie Mellon CyLab – *Management and Education on the Risk of Insider Threat (MERIT)*
- PERSEREC

# CERT/USSS *Insider Threat Study*

Definition of insider:

*Current or former employees or contractors who*

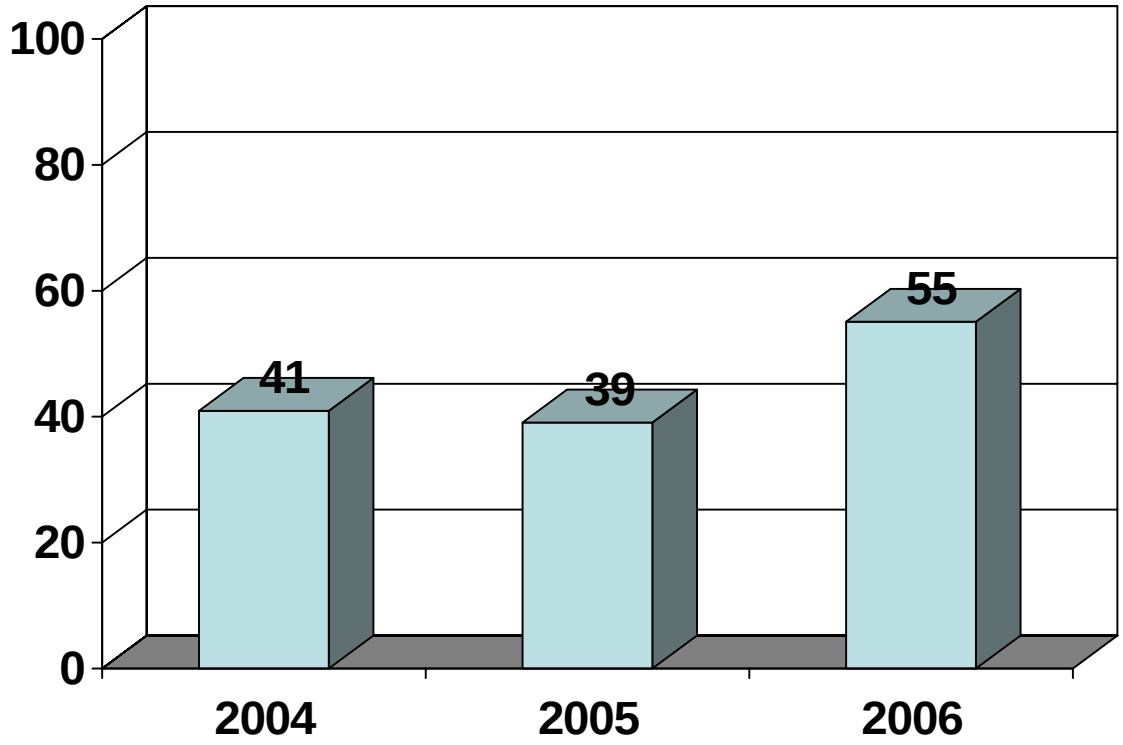
- *intentionally exceeded or misused an authorized level of access to networks, systems or data in a manner that*
- *targeted a specific individual or affected the security of the organization's data, systems and/or daily business operations*


# 2006 e-Crime Watch Survey

CSO Magazine, USSS & CERT

434 respondents

**Percentage of Participants Who Experienced an Insider Incident**



# *Insider Threat Study*

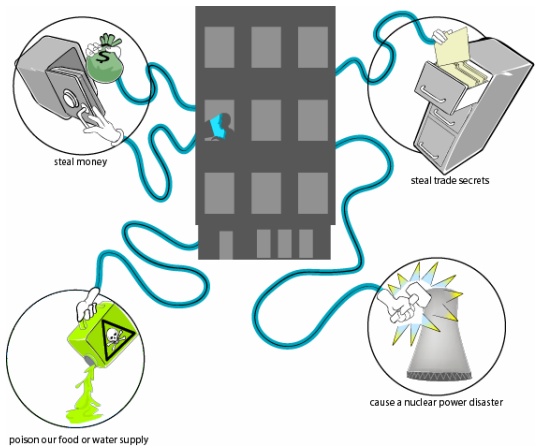
Funded by US Secret Service (partially by Department of Homeland Security)

Examined technical & psychological aspects

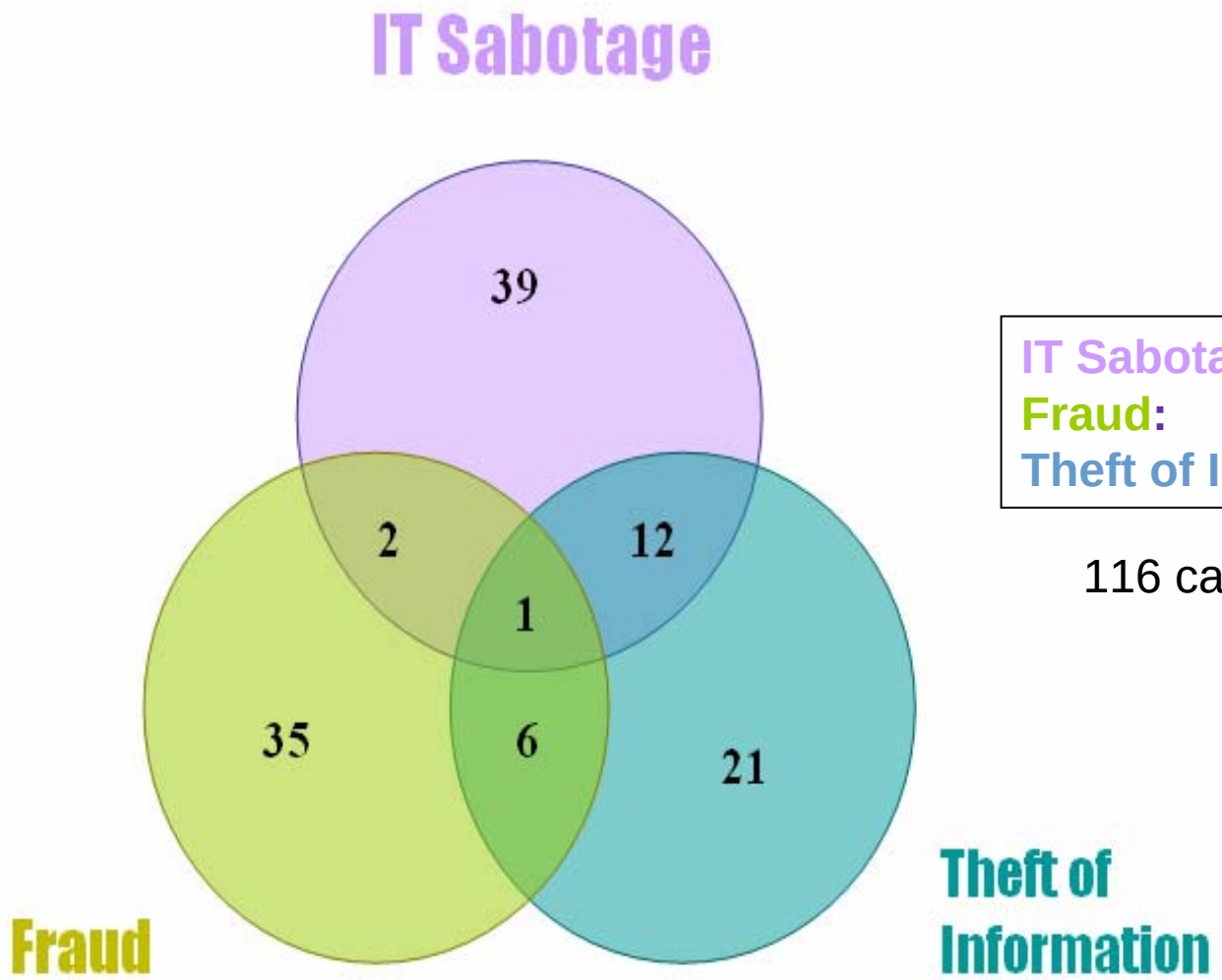
Analyzed actual cases to develop information for prevention & early detection

Methodology:

- Collected cases (150)
- Codebooks
- Interviews
- Reports
- Training



# Insider Threat Case Breakdown




| | |
|---|---|
| IT Sabotage: | 54 |
| Fraud: | 44 |
| Theft of IP: | 40 |

116 cases total

# *Insider Threats During the SDLC*

# Phases of the Life Cycle Exploited

Requirements definition

System design

System implementation

System deployment

System maintenance

# Examples of Impacts

Company went out of business

Fraud losses up to $691 Million

Drivers licenses created for individuals who could not get a legitimate license

Disruption of telecommunications services by telecom firm

Court records, credit records, and other critical data modified

Virus planted on customers' systems

# Requirements Definition Oversights

Neglecting to define **authentication** and **role-based access control** requirements simplified insider attacks.

Neglecting to define **security requirements/separation of duties** for **automated business processes** provided an easy method for insider attack.

Neglecting to define requirements for **automated data integrity checks** gave insiders the security of knowing their actions would not be detected.

# Case Examples – Requirements Definition

EXAMPLE#1

195 illegitimate drivers licenses are created and sold by a Police Communications Officer who accidentally discovers she can create them.

Insider THREAT

EXAMPLE#2

A system administrator deletes 18 months of cancer research after being fired since no electronic access controls stood in his way.

# System Design Oversights

Insufficient attention to security details in **automated workflow processes** enabled insiders to commit malicious activity.

Insufficient **separation of duties** facilitated insider crimes.

- not designed at all
- no one to “check the checker”

Neglecting to consider security vulnerabilities posed by “**authorized system overrides**” resulted in an easy method for insiders to ”get around the rules”.

# Case Examples – System Design

EXAMPLE#1

Special function to expedite handling of cases allows two case workers to pocket $32,000 in kickbacks.


EXAMPLE#2

An employee realizes there is no oversight in his company's system & business processes, so he works with organized crime to enter & profit from $20 million in fake health insurance claims.

# System Implementation Exploits

Lack of **code reviews** allowed insertion of "backdoors" into source code.

Inability to **attribute actions** to a single user enabled a project leader to sabotage his own team's development project.

# Case Examples – System Implementation

EXAMPLE#1

An 18 year old former web developer uses backdoors he inserted into his code to access his former company's network, spam their customers, alter their applications, and ultimately put them out of business.

EXAMPLE#2

A project leader for a software project sabotages his own project rather than admit to inability to meet project deadlines.

# System Deployment Oversights

Lack of enforcement of **documentation practices** and **backup procedures** prohibited recovery efforts when an insider deleted the only copy of source code for a production system.

Use of the same **password file** for development and the operational system enabled insiders to access and steal sensitive data from the operational system.

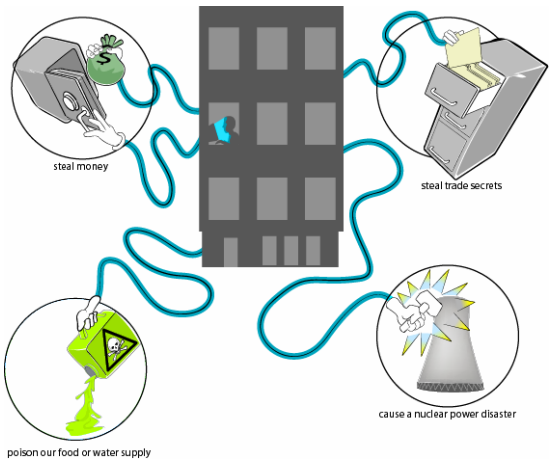
**Unrestricted access** to all customers' systems enabled a computer technician to plant a virus directly on customer networks.

Lack of **configuration control** and well-defined **business processes** enabled libelous material to be published to organization's website.

# Case Examples – System Deployment

EXAMPLE#1

A computer technician uses his unrestricted access to the company's customers' systems to plant a virus on the customers' networks that brings their systems to a halt one morning.


EXAMPLE#2

A software engineer deliberately creates no documentation or backups for his source code, then deletes the only copy of the source code once the system is in production.

# System Maintenance Issues

Lack of **code reviews** facilitated insertion of malicious code.

Ineffective **configuration control** practices enabled release of unauthorized code into production.

Ineffective or lack of **backup processes** amplified the impact of mass deletion of data.

**End-user access** to source code for systems they used enabled modification of security measures built into the source code.

Ignoring known **system vulnerabilities** provided an easy exploit method.

# Case Examples – System Maintenance

EXAMPLE#1

A foreign currency trader covers up losses of $691 million over a 5 year period by making unauthorized changes to the source code.



EXAMPLE#2

A logic bomb sits undetected for 6 months before finally wreaking havoc on a telecommunications firm.

# Summary – Most Prevalent SDLC Issues

IT Sabotage:

- System architecture that allows for efficient recovery or sustains the organization during disasters
- Configuration and access control of source code
- Formal code review/inspection to prevent malicious code from being inserted into production applications

Fraud:

- Existence and enforcement of authorization/approval steps in automated work flow to ensure proper approvals for critical business functions

Theft of Sensitive or Confidential Information:

- Configuration and access control of source code

# Best Practices

# CyLab Common Sense Guide - Best Practices

Institute periodic enterprise-wide risk assessments.

Institute periodic security awareness training for all employees.

Enforce separation of duties and least privilege.

Implement strict password and account management policies and practices.

Log, monitor, and audit employee online actions.

Use extra caution with system administrators and privileged users.

Actively defend against malicious code.

Use layered defense against remote attacks.

Monitor and respond to suspicious or disruptive behavior.

Deactivate computer access following termination.

Collect and save data for use in investigations.

Implement secure backup and recovery processes.

Clearly document insider threat controls.

# Points of Contact

**Insider Threat Team Lead:**
Dawn M. Cappelli
Senior Member of the Technical Staff
CERT Program
Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213-3890
+1 412 268-9136 – Phone
dmc@cert.org – Email

**Business Development:**
Joseph McLeod
Business Manager
Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213-3890
+1 412 268-6674 – Phone
+1 412-291-3054 – FAX
+1 412-478-3075 – Mobile
jmcleod@sei.cmu.edu – Email

http://www.cert.org/insider_threat/