The Pennsylvania State University

The Graduate School

College of Information Sciences and Technology

TOWARDS FLEXIBLE AND REALISTIC INSIDER MISSION SIMULATION

A Thesis in

Information Sciences and Technology

by

Tao Zhang

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science

August 2018

The thesis of Tao Zhang was reviewed and approved* by the following:

Peng Liu
Professor of Information Sciences and Technology
Thesis Adviser

Sencun Zhu
Associate Professor of Information Sciences and Technology

Dinghao Wu
Associate Professor of Information Sciences and Technology

Mary Beth Rosson
Professor of Information Sciences and Technology
Associate Dean for Graduate Programs

* Signatures are on file in the Graduate School.

ABSTRACT

With the widespread application of information technology, organizations rely more and more on networked information system to manage their daily affairs. As a result, modern organizations are increasingly vulnerable to insider threat. Insider incidences happen more and more frequently and cause significant losses. Consequently, insider attacks have become a growing concern in security area. Lacking real world insider threat data, researches in insider threat have been seriously constrained. In this paper, we are going to introduce a simulation framework to help simulate organizational behavior with insider mission performed internally. With simulated insider mission, we are able to generate insider threat data based on the event logs of our simulator. In addition to event log, we can provide all the ground truth information regarding the malicious insider, intranet system and organization. In the paper, we will also present and discuss the measures taken to achieve high-fidelity insider threat data. In addition, we construct insider mission simulator to be flexible, offering various insider mission scenarios and attacking strategies. Moreover, we are going to incorporate varieties of obfuscation techniques into insider mission simulation. In this way, we can easily generate diverse insider data sets to support test and validation for intrusion detection systems.

TABLE OF CONTENTS

List of Tables

List of Figures

Acknowledgement

This Thesis is supported by DARPA Cyber Insider (CINDER, FA8750-11-C-0038) program.

Chapter 1 Introduction

Nowadays, organizations, such as companies and governments, increasingly rely on information technology to manage their daily affairs and improve working efficiency, which in turn makes them more and more vulnerable to insider threats [1]. By insiders, we mean the legitimate users with elevated privileges and knowledge about internal system and security mechanisms [2], which may include employees, contractors, business partners and even clients.

According to the E-Crime watch survey [3], which was jointly conducted by the CERT Coordination Center (CERT/CC), the U.S. Secret Service, and other organizations in 2007, 31% of the electronic crimes were attributed to insiders. In addition, 49% of polled security specialists and law enforcement officials have experienced malicious insider incidents within a year [4]. Moreover, insider attacks are far more destructive compared with traditional attacks. A survey showed that more than 30% of financial insider incidents caused losses exceeding $500,000 each [5].

Insider threat has become a growing concern in security area. However, insider threat research has been severely constrained by a lack of real world insider data [6]. There are two major challenges in getting insider data:

- Real insider mission data is too sensitive to share. Insider mission data usually contains sensitive information, such as organization's security policies, sensor deployment, firewall configurations, etc. In addition, system log and network sensor log may also include organization's business information, some of which are of high confidentiality. For example, financial information, next year's business development plan and critical product workflows are business secretes to enterprises. Moreover, to collect real insider data, all the network events should be logged and all the network traffic should be monitored. Release of such kind of data set may lead to privacy issues. Therefore, organizations are reluctant to share insider mission data with the research community.

- Not all ground truth is known/flagged in real data. Another problem with real world insider data is that we cannot get to know all the ground truth, which is critical to initial research. For example, the data set may not provide the user information (e.g. employee id, job responsibility, access account, etc.) or asset information (e.g. web server architecture, web application information, file system architecture, file information, etc.). In addition, we cannot tell which audit logs are generated by insider mission behavior, especially when the mission is at an early stage. Actually, this information is very important to study and analyze the malicious insider's motivation and attacking strategy.

In addition to the challenges to obtain insider data, real world insider data also has its inherent limitations. Real insider data is what it is, which is to say, one insider incident only presents one insider mission scenario and one insider behavior pattern, which is far from enough for research community. However, getting varieties of real insider data sets is challenging.

In this paper, we are going to investigate into the research question: how to achieve flexible and realistic insider mission simulation to better support insider threat research. To be more specific, we want to investigate various business processes in different types of organizations, regular behavior models for different user roles, widely used attacking strategies by malicious insiders. In addition, we are also interested in providing extensive flexibility regarding insider mission happening in different contexts (organization environment), different combinations of evasion strategies, different insider knowledge set, skill set, as well as privileges.

Due to the challenges in obtaining real world insider data, we choose to simulate insider mission process and generate the insider data set. Although simulated data may be biased and different from real world cases, the generated data set has its advantages: 1) we can provide or flag all the ground truth in the data set; 2) we can configure the system parameters to provide different insider mission scenarios and insider behavior patterns. While we admit the shortcomings and limitations of simulated data, it does help facilitate insider threat research.

Chapter 2 Related Works

Currently, there are mainly three existing insider threat data sets commonly used by the research community. Greenberg [7] collected a corpus of Unix command-line data, and Maxion [8] assembled it into an insider threat data set by replacing insider commands with benign commands and sanitizing private information. In Greenberg data set, each account command is enriched with flags and arguments. Lane and Brodley [9] also collected users' commands with similar substitution and sanitization measures. Another popular data set used for insider threat research is developed by Schonlau et al [10]. They collected the names of programs executed instead of the full command line and use real world commands of normal accounts to simulate the command list that contain malicious insiders' activities. Some commands are injected as insider commands into victims' commands list. It simulates that some accounts are compromised and the attackers manage these accounts to do some malicious activities. As long as the behavior patterns of the account owners and the attackers are different, it is possible to detect the malicious activities. However, some research has shown Schonlau data set was not suitable for insider attack detection [11]. One reason is that insider actions might often be intentionally obfuscated, which cannot be simulated in Schonlau data set.

Due to a lack of insider threat data, some researchers simulate account behaviors and generate accounts command data with insider attacks, to facilitate insider detection research. Chinchani et al. developed RACOON, which models customizable profiles and generate account commands based on the profiles [12]. Usim is a similar behavior simulation and command generation system [13]. People can extract features from real account behavior data set and use these features to build profiles for Usim or RACOON, which ensures fidelity of the generated data to some extent.

As we can see, all the insider threat data sets above contain only command-line data, which is not always the case in real world situation. Command line is not always the interface between users and information systems. The real world enterprise network is a complex environment, in which you may not always be able to record user behavior by logging the command-line data. There are also many other types of network events that we

have to take advantage of other tools to keep track of them, such as the Apache web server log, Snort or other network sensors. To fill in the gap of lacking hybrid insider threat data, we propose a flexible and realistic insider mission simulation framework that can generate insider data with varieties of network events.

Chapter 3 Approach

3.1 Overview of methodology

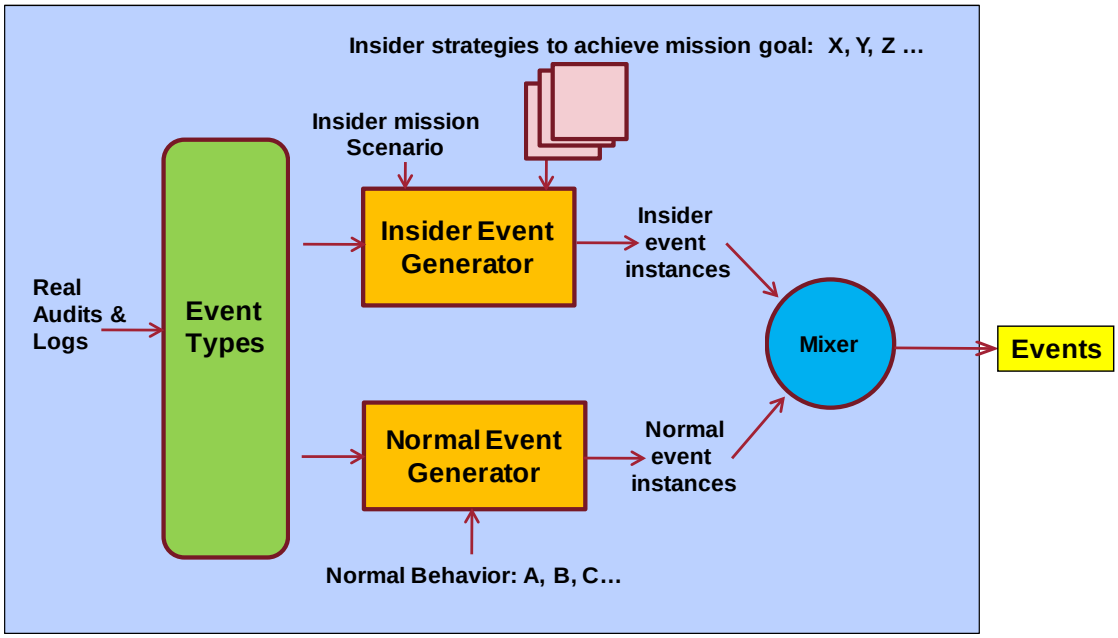

Figure 3.1 Insider Mission Simulation Framework

Figure 3.1 shows the composition of our insider mission simulation framework. More specifically, our approach to simulate the insider mission process is as follows:

a) Before a synthesized insider mission scenario data set is generated, we firstly define the relevant event types and activity types based on real world sensors and observables. To be specific, we analyze the real log data from various sensors (operating system log, network monitoring tools, intrusion detection system, etc.) and extract event types with attributes from real audit logs.

b) Based on the insider mission scenarios provided by red team, we construct the 5-layer insider mission hierarchical state machine. In particular, we divide the mission progress into 5 levels: mission level, dimension level, activity level, business process level and event level. Mission-level progress is the overall accomplishment

of the insider mission and event-level progress is the finest granularity of mission progress. The five layers form up a mission hierarchy. We define each layer and build both the intra-layer state machine and inter-layer state machine, forming up a hierarchical state machine, which guides the insider mission data generation.

c) Based on the investigation into the business process in a workflow-oriented organization, we figure out typical user roles in the organization, such as database administrator, security analyst, application developer, etc. Based on the investigation, we figure out the job responsibility for each user role and construct the hierarchical state machine accordingly. These hierarchical state machines will guide background data generation.

d) Before simulation starts, predetermine all ground truth, including asset information (e.g. enterprise network topology, server information), user profile information (e.g. user id, job title) and behavior models for different user roles. Behavior models are built based upon corresponding hierarchical state machines constructed and behavior models, as well as the hierarchical state machines, are hided from detection algorithms. In other words, we configure all the details of the organization as well as user behavior models.

e) Insider mission events and background events are simulated concurrently and in a time sequential way. In simulation, each user or software agent is represented by an individual thread and the user behavior or agent behavior is guided by the behavior models and constrained by the current system state. A thread scheduler is built to manage the concurrency and time sequence. System log is maintained to log all the event instances that have happened.

f) After the simulation terminates, data generator output the system logs in XML format.

3.2 Behavior model

To simulate insider mission, we define the insider mission scenario and the data abstraction hierarchy, shown in Figure 3.2, that consists of “mission”, “dimension”,

"business process" and "event" layer. Based on the data hierarchy, we design our hierarchical state machines, which play the role of behavior models, guiding either normal users' behavior or automated agents' behavior.

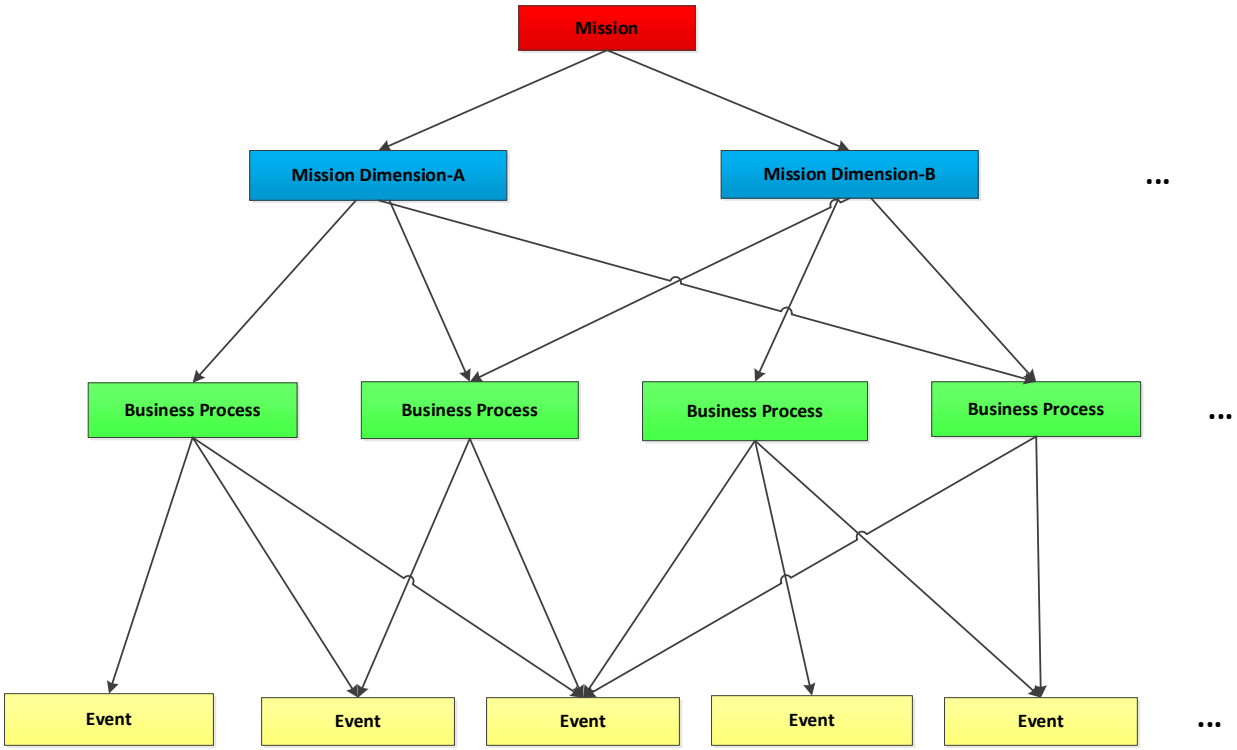

Figure 3.2 Data Abstraction Hierarchy

The data abstraction hierarchy is divided into four layers: insider mission, mission dimension, business process and event. On top of the hierarchy is the insider mission, consisting of a set of mission dimensions. The business process layer is closely related to the proposed insider mission scenarios: each dimension involves several business processes and one business process can be shared by several dimensions. At the bottom of the hierarchy, lies the event layer, which is the finest granularity in our data abstraction. Each event instance, belonging to a certain event type, can be shared by different business processes. Event type should be neutral – regarding whether they look like part of the insider mission or normal organizational behavior.

Algorithm 1. Hierarchical State Machine Algorithm
Input: thread information, current state, state transition probability tables.
Output: execute an event and transit to the next state according to the state transition rules.

```
1. Load (current_state.activity);
2. Call (current_activity());
        2.1 Load (current_state.business_process);
    2.2 Call (current_state. business_process());
        2.2.1   Load (current_state.event);
        2.2.2   Call (current_state.event());
            2.2.2.1  Execute (current_state.event());
            2.2.2.2  Log (event);
            2.2.2.3  Update_time (thread.time);
        2.2.3   Load (thread.event_transition_pro_table);
        2.2.4   Next_event (current_state.event);
    2.3 If (Isfinished (current_state.business_process) = = true){
        2.3.1   Load (thread. business_process_transition_pro_table);
        2.3.2   Next_ business_process(current_state.business_process);
        }
3. If (Isfinished (current_state.activity) = = true){
    3.1 Load (thread.activity_transition_pro_table);
    3.2 Next_activity (current_state.activity);
    }
```

Algorithm 1 above is a three-layer hierarchical state machine algorithm designed to simulate user behavior (either normal users' behavior or automated agents' behavior) according to predetermined behavior model. Here, the state transition probability tables play the role of behavior models, guiding user behavior. To be specific, we have three levels of state transition probability tables: activity level, business process level and event level. These three levels of probability tables, together with pre-designed data hierarchy, constitutes the three-layer hierarchical state machine, including both inter-layer state machine and intra-layer state machine.

3.3 Business processes

In our approach, we developed a set of business processes to help control the insider mission evolvement and the normal workflow of the organization. There are mainly three types of business processes in our simulation system: insider mission specific business processes, operational business processes and supporting business processes.

Insider mission specific business processes intend to perform the insider mission and achieve the mission goal. Operational business processes are related to the organization's business. Their purpose is to achieve the organization's business goal. They are the major workflow in the organization. Supporting business processes are those that are carried out by the supporting department like human resource, accounting, etc. They work to support the organization's operation.

Insider mission utilizes both insider mission specific business processes and a portion of operational business processes to achieve mission goal. It is this feature makes insider threat difficult to tackle. Operational business processes and supporting processes are benign and they produce the organizational daily activities. They serve as the normal background events in our simulation.

3.3.1 Insider mission specific processes

Although malicious insiders try to rely on existing business processes to achieve insider mission, it is not always feasible to do this. In this section, we present some example insider mission specific business processes:

M1 Monitor data in the database

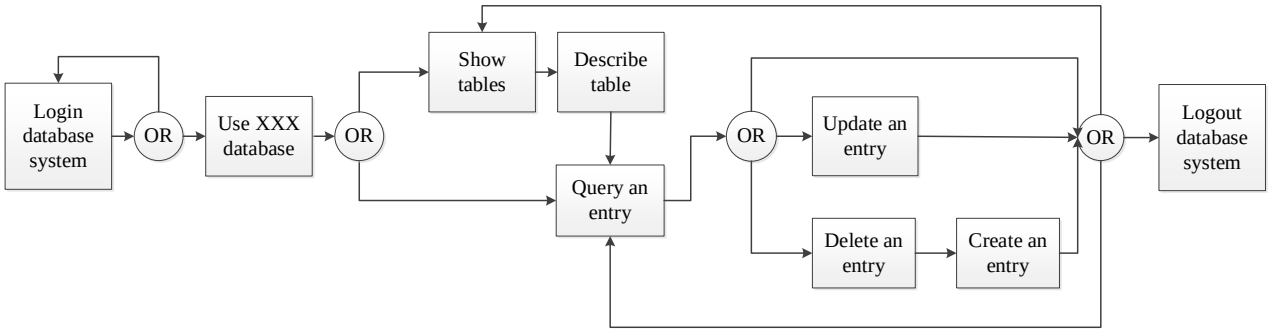

Figure 3.3 Bsiness Process M1 Flow Diagram

M2 Monitor data in the file system

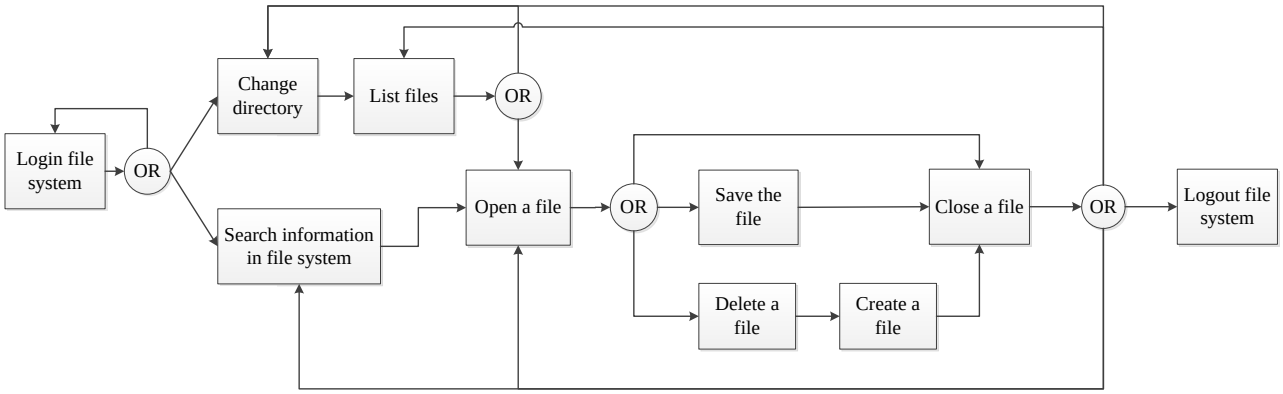

Figure 3.4 Bsiness Process M2 Flow Diagram

M3 Monitor data via the web UI

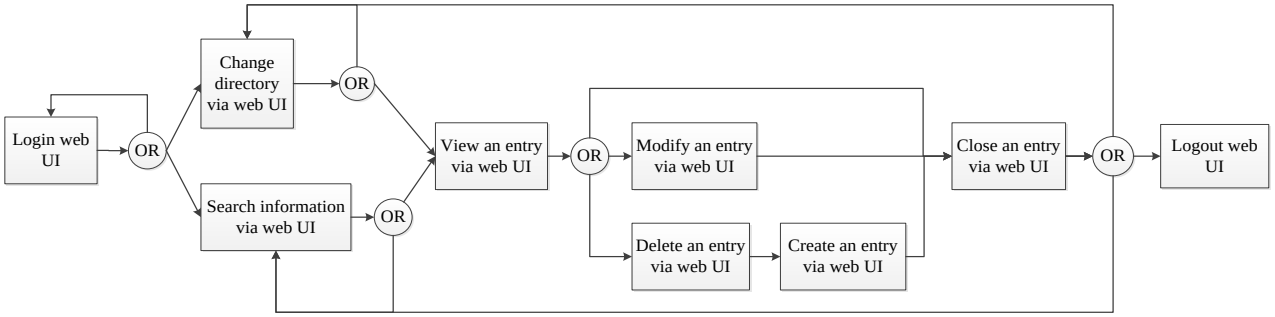

Figure 3.5 Bsiness Process M3 Flow Diagram

3.3.2 Operational business processes

There exists a set of regular business processes in an organization for its commercial activities. A portion of operational business processes may be utilized by malicious insiders to achieve insider mission. In this section, we present some example operational business processes:

O001 Information collection

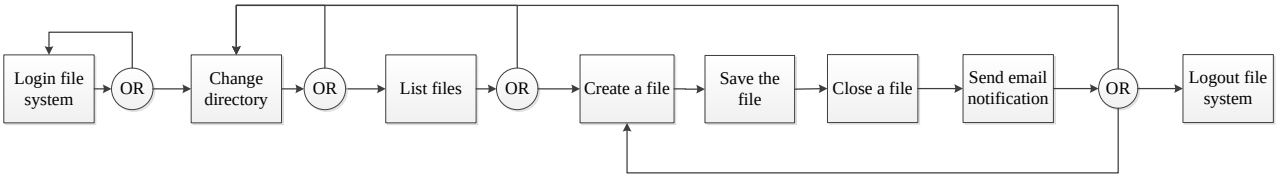

Figure 3.6 Bsiness Process O001 Flow Diagram

O002 Send an email

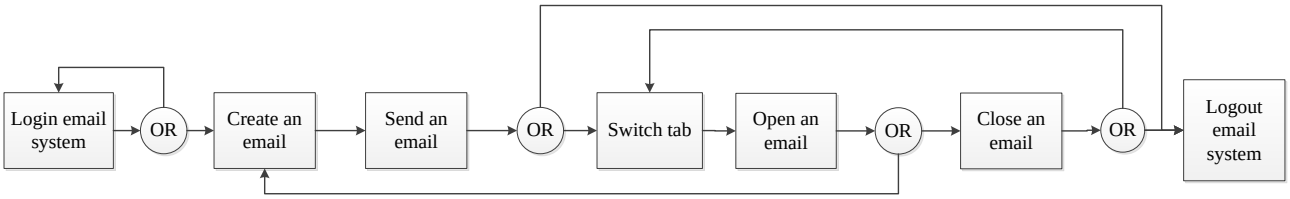

Figure 3.7 Bsiness Process O002 Flow Diagram

O003 Initial analysis

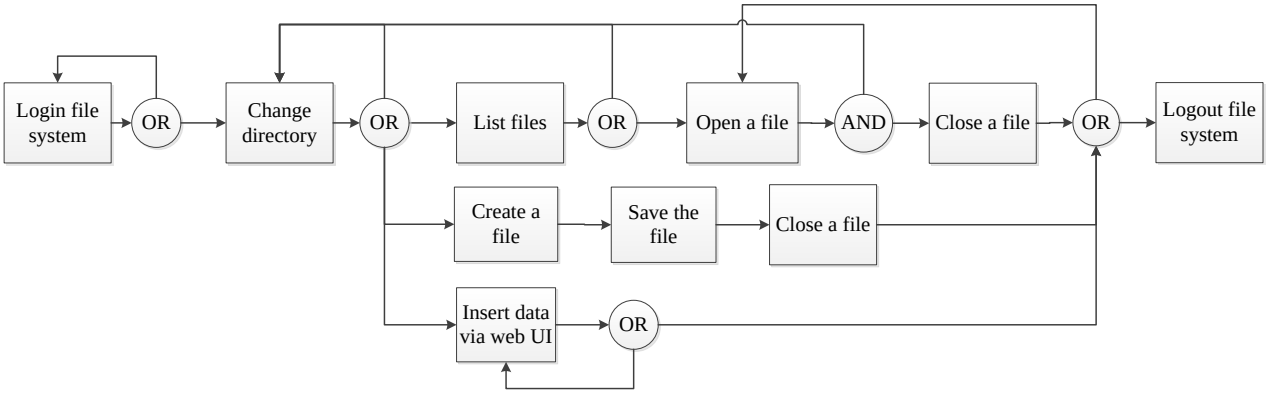

Figure 3.8 Bsiness Process O003 Flow Diagram

O004 Information update

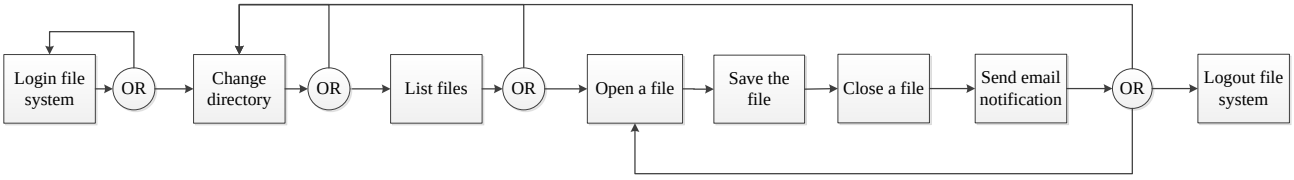

Figure 3.9 Bsiness Process O004 Flow Diagram

O005 Generate report

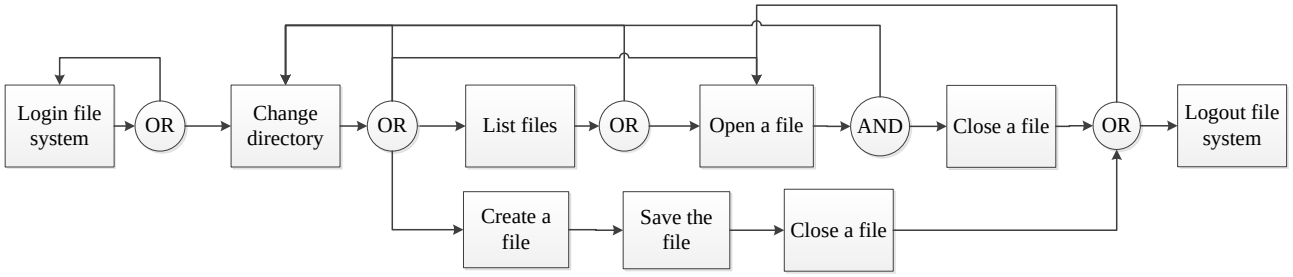

Figure 3.10 Bsiness Process O005 Flow Diagram

O101. Search for intelligent analysis process/policy on web server

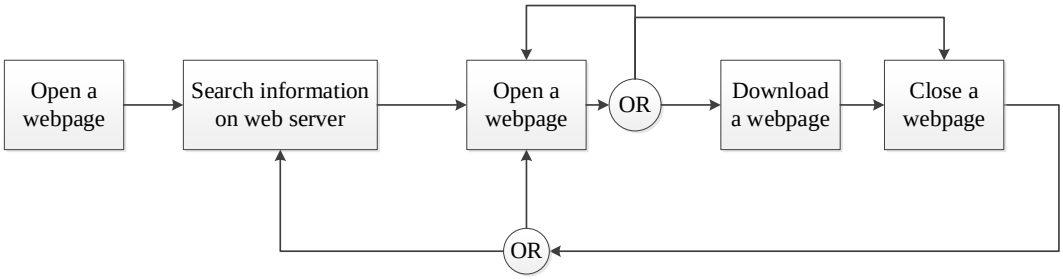

Figure 3.11 Bsiness Process O101 Flow Diagram

O102 Search for intelligent analysis process/policy in file system

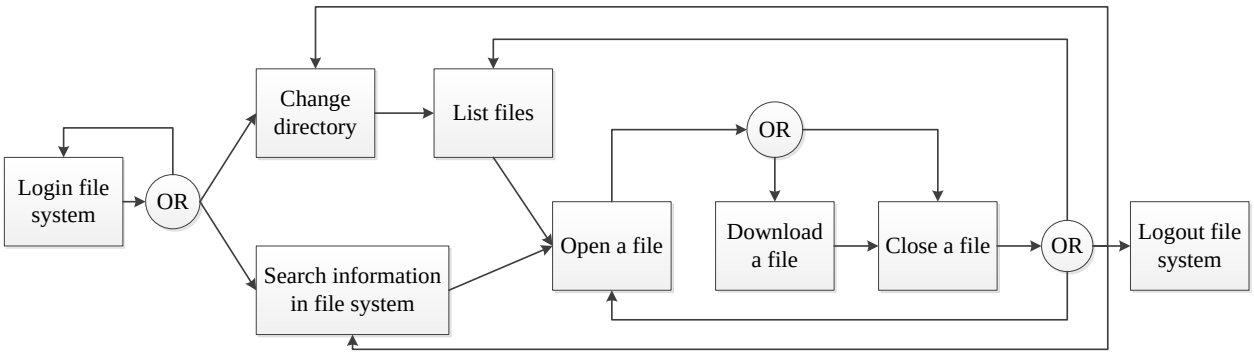

Figure 3.12 Bsiness Process O102 Flow Diagram

O103 Search for mission critical data in the database

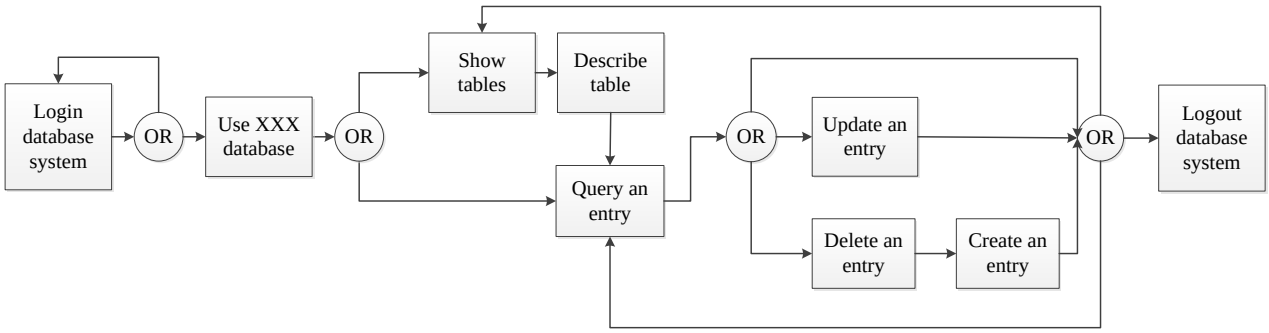

Figure 3.13 Bsiness Process O103 Flow Diagram

O104 Modify data on the database server

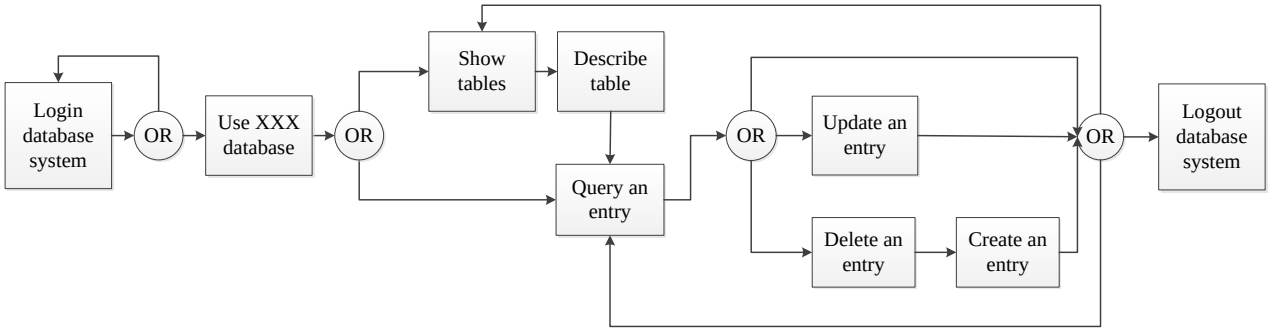

Figure 3.14 Bsiness Process O104 Flow Diagram

O105 Search mission critical data in the file system

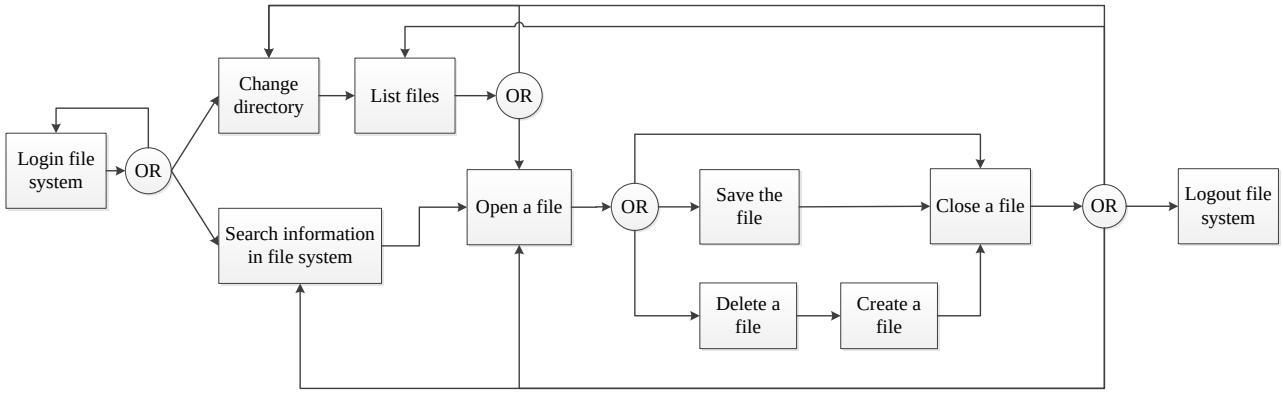

Figure 3.15 Bsiness Process O105 Flow Diagram

O106 Modify data in the file system

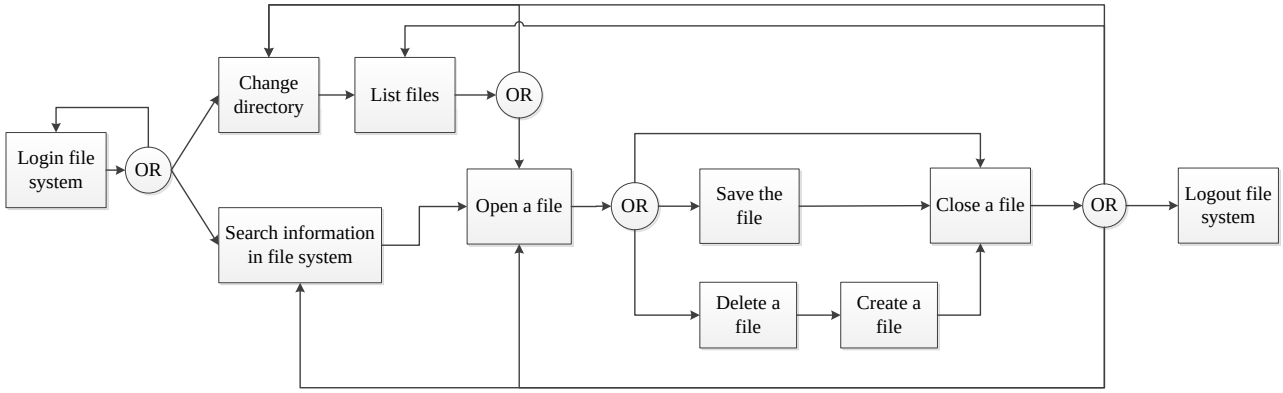

Figure 3.16 Bsiness Process O106 Flow Diagram

O107 Search mission critical data via Web UI

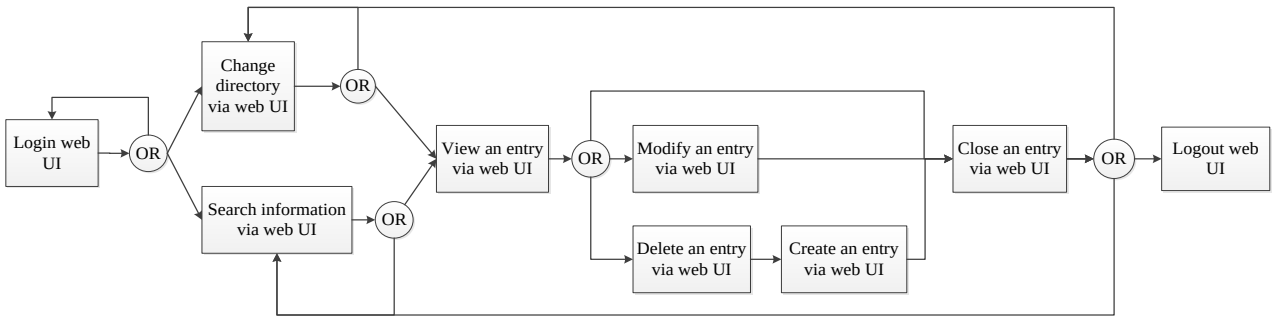

Figure 3.17 Bsiness Process O107 Flow Diagram

O108 Modify data via Web UI

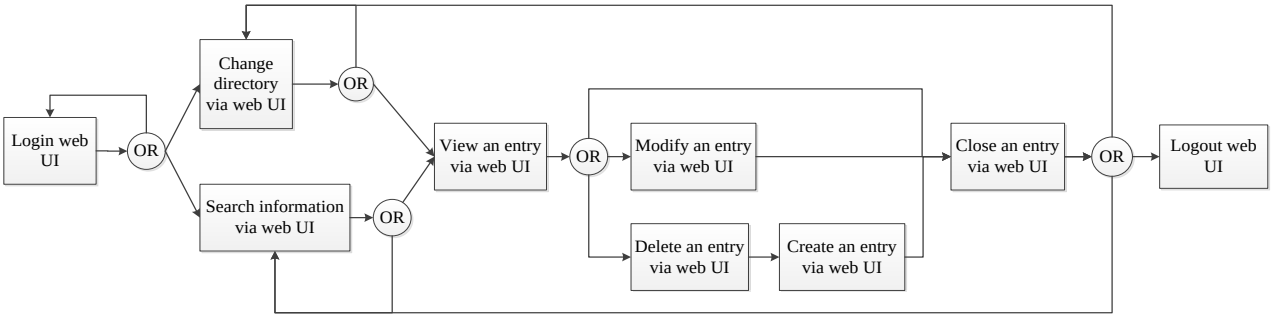

Figure 3.18 Bsiness Process O108 Flow Diagram

3.3.3 Supporting business processes

There exist varieties of background business processes in an organization to support its commercial activities. In this section, we present some example supporting business processes:

S1 Recruitment (new employee registration)

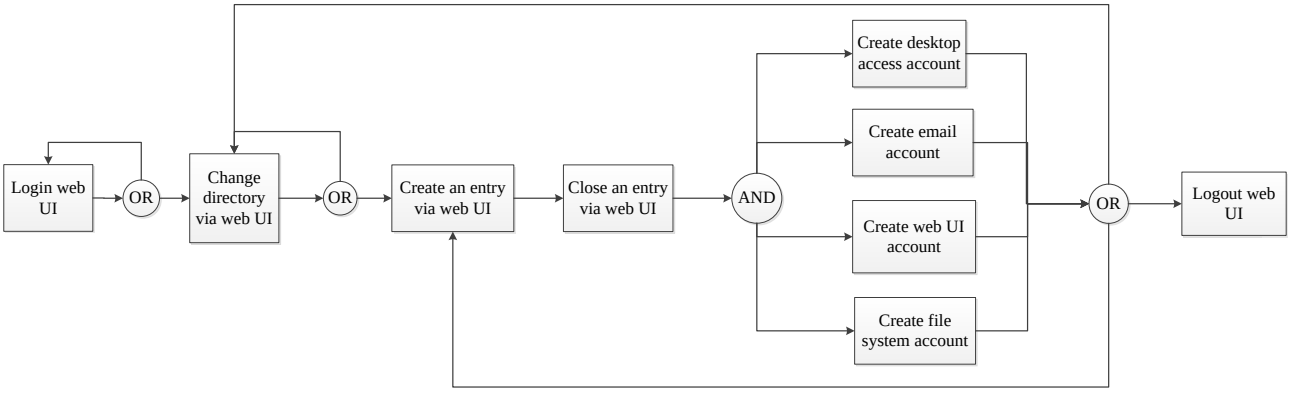

Figure 3.19 Bsiness Process S1 Flow Diagram

S2 Training (tutorial (the doc is from initial analysis) (read only)) + quiz (read+write))

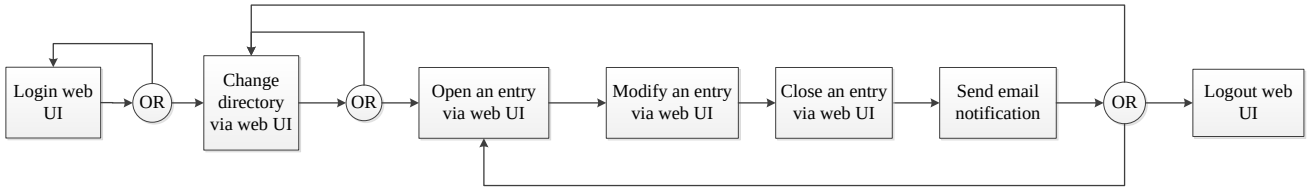

Figure 3.20 Bsiness Process S2 Flow Diagram

S3 Annual review

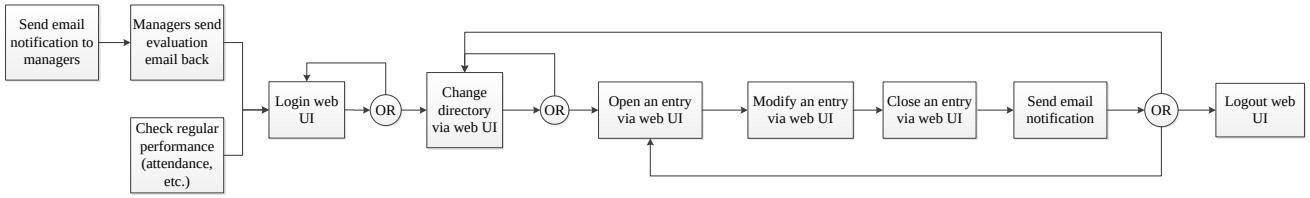

Figure 3.21 Bsiness Process S3 Flow Diagram

S4 Promotion

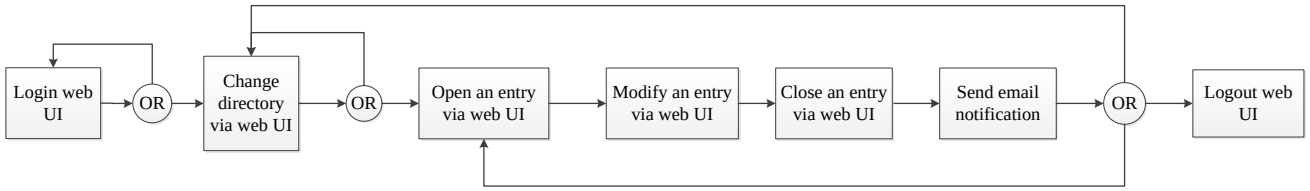

Figure 3.22 Bsiness Process S4 Flow Diagram

S5 Layoff

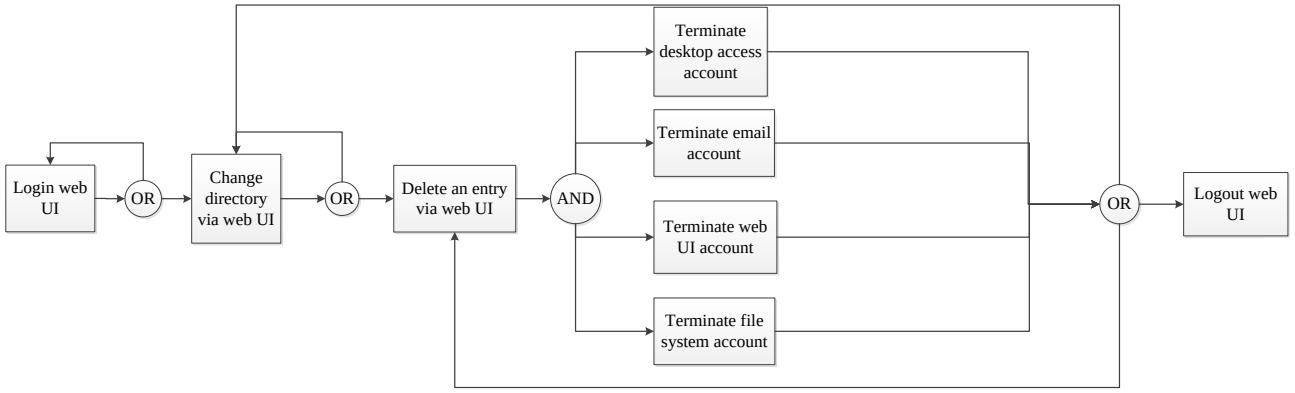

Figure 3.23 Bsiness Process S5 Flow Diagram

S6 Adjust salary (increase or decrease)

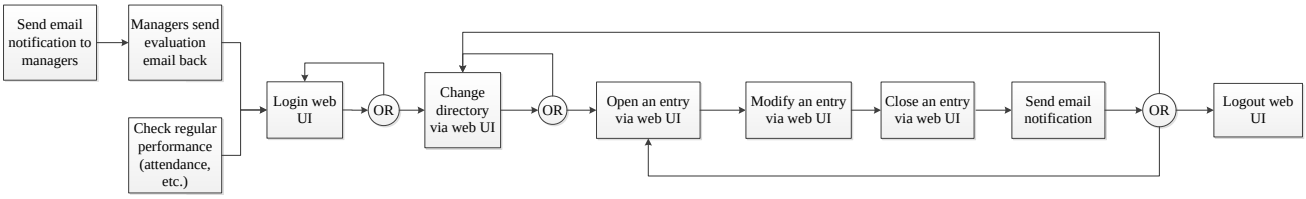

Figure 3.24 Bsiness Process S6 Flow Diagram

3.4 Insider mission design

In this section, we are going to present the insider mission design at dimension level and business process level.

3.4.1 Insider mission dimensions

With rich experience in real world penetration and testing, the Boeing Red Team provides their expertise and helps us to develop the insider mission dimensions. According to the Boeing Red Team, insider mission can be divided into 6 dimensions:

A. Reconnaissance: it is also known as information gathering. That is to help the malicious insiders to understand how the list is collected, analyzed, reported, reviewed, stored, updated, and accessed.

B. Tamper data in the database: a portion of sensitive data is stored in the database. Malicious insiders would like to find those mission critical data and tamper it in the database.

C. Tamper data on the file server: all the files are stored on the file server. Malicious insiders are concerned about the mission critical files. They can tamper mission critical data on the file server.

D. Tamper data through the web UI: a portion of sensitive data can also be accessed and modified through the web UI with certain privilege. Malicious insiders can tamper the mission critical data via the web UI.

E. Watch for data updates: in order to get the latest mission critical data updates, malicious insiders would examine the mission critical data in the database, in the file system or through the web UI periodically. They can take advantage of automated tools such as ‘inotify’ for file system update monitoring. Periodically issue SQL queries and file listing commands using a user script.

F. Avoid detection: in order to protect themselves and continue mission operations, malicious insiders would like to avoid detection. They tend to use different strategies and tools to cover their traces. An automated tool such as TouchPro allows changes on file time attributes: folder date, timestamp, etc.

3.4.2 Dimension interrelationships

The mission Dimensions can be categorized into three stages: 1) Reconnaissance stage, 2) Tampering stage, and 3) Cover-up stage. The reconnaissance stage consists of the first dimension, Dimension A. The tampering stage consists of database tampering (Dimension B), file system tampering (Dimension C), and web UI based tampering (Dimension D). These three tampering dimensions are not necessarily redundant and may not need to be performed in certain order: key information is stored in the database; supplementary information may be stored in the file systems; web UI provides means to access both data sources. The cover-up stage consisting of Dimensions E and F, is to accomplish continued surveillance and mission operation. In order for the adversary to continue monitor List update (Dimension E), he/she should remain undetected (Dimension F).

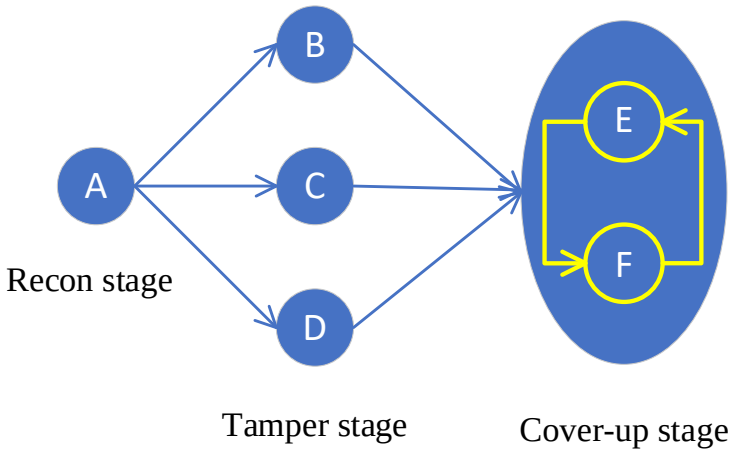

Figure 3.25 Dimension Interrelationships

3.4.3 Insider mission business processes

Based on the mission Dimensions design, we developed the insider mission business processes accordingly. Each Dimension employs a couple of business processes to accomplish part of the mission goal.

Dimension A involves business processes O101 and O102:

- O101 Search for intelligent analysis process/policy on web server
- O102 Search for intelligent analysis process/policy in file system

Dimension B is involved with O103 and O104:

- O103 Search for mission critical data in the database
- O104 Modify data on the database server

Dimension C consists of O105 and O106:

- O105 Search for mission critical data in the file system
- O106 Modify data in the file system

Dimension D includes O107 and O108:

- O107 Search for mission critical data through the web UI
- O108 Modify data via Web UI

Dimension E involves M1, M2 and M3:

- M1 Monitor data in the database
- M2 Monitor data in the file system
- M3 Monitor data via the web UI

Dimension F is to avoid detection. Detection evasion strategies are applied to the insider mission business processes directly. No business process or event instance is visible.

3.4.4 Simulated insider mission instances

Here, we describe one simulated insider mission instance:

1) the malicious insider continuously searches for intelligent analysis process/policy on web server (business process O101), go to step 2) or 3);
2) the malicious insider continuously searches for intelligent analysis process/policy in file system (business process O102), go to step 1) or 3);
3) the malicious insider continuously searches for mission critical data in the database (business process O103), go to step 4);
4) the malicious insider modifies mission critical data in the database (business process O104), go to step 3) or 5);
5) the malicious insider continuously searches for mission critical data in the file system (business process O105), go to step 6);
6) the malicious insider modifies mission critical data in the file system (business process O106), go to step 5) or 7);
7) the malicious insider continuously searches for mission critical data through web UI (business process O107), go to step 8);
8) the malicious insider modifies mission critical data in the file system (business process O108), go to step 7) or 9);
9) the malicious insider keeps monitoring mission critical data in the database (business process M1), go to step 4), 10) or 11);
10) the malicious insider keeps monitoring mission critical data in the file system (business process M2), go to step 6), 9) or 11);
11) the malicious insider keeps monitoring mission critical data through the web UI (business process M3), go to step 8), 9) or 10);
12) the malicious insider keeps using detection evasion strategies while working in the previous steps to avoid detection.

3.5 Concurrency control mechanism

In the original prototype, we adopt multi-threaded simulation to manage the concurrency of insider events and normal-user events. To be specific, we create each thread to represent one user, either the insider or a normal user, and then this thread will simulate the user's behavior according to the pre-determined behavior model. When the simulator runs, each thread will be initialized and triggered to run concurrently. In other words, we let the operating system take control of concurrency.

As the simulation scales, the concurrency control mechanism turns out to be inefficient. To improve the efficiency, we come up with the idea of event-driven simulation. Here, we make an assumption that event instance is the smallest granularity in our simulation. That is to say, event cannot be further divided into several actions/operations in our simulation environment. Originally, we take advantage of the scheduling service provided by the operating system to manage the concurrency, which is relatively slow. Now, we build a scheduler by ourselves to take control of the concurrency management, speeding up the simulation significantly (by more than 100 times when the simulation scales).

Algorithm 2 below is a thread-level algorithm designed to help manage concurrent simulation. Here, a thread is a hierarchical state machine representing either a normal user or an automated agent. The thread list is the whole list of threads, including normal users and automated agents, ordered in a time-sequential way. A thread has two states: active and inactive, indicating whether this thread is able to generate events or not. Running a thread at a time only executes one event, and then the whole system will wait for re-scheduling to maintain the time order.

Algorithm 2. Concurrent Simulation Algorithm
Input: thread (state machine) list, system time, simulation ending time.
Output: let each thread (state machine) progress in a concurrent and sequential way.

```
While (system_time < ending_time) {
    1. While (system_time<Earlieast_thread_time))      //when system time hasn't reached the
       next
            Clock_tick (system_time);                  // event time, update system time
    2. Execute (threadlist->Earliest_thread);         //threadlist is the whole list of thread ordered
    3. Reschedule threadlist                           //in a time-sequential way
        If (threadlist->Earliest_thread.active = = false) {
            Schedule (Earliest_thread) at the end of thread list;
            Schedule (Earliest_thread->next_thread) as the Earliest_thread;
        }
        Else if (IsEarliest(Earliest_thread) = = false) {
            Reschedule (Earliest_thread) to maintain the time sequence of thread list;
            Schedule (Earliest_thread->next_thread) as the Earliest_thread;
        }
}
```

In addition to the traditional insider mission simulation, we provide an alternative choice - automated-agent-centric insider mission simulation – which is the future trend of insider threat. In the future, skillful attackers will take advantage of advanced software agents to launch attacks, achieving the mission goal.

Software agents have advantages over human beings that they can be very patient and persistent while conducting insider mission. Human beings usually cannot afford the time and energy to lower the pace to avoid detection.

To accommodate the future trend, we support automated-agent-centric simulation by designing a set of automated agents (according to investigation into malware):

- Reconnaissance Agent: mainly collect relevant information to support insider mission process, such as organization's IT security policies, infrastructure information, business process, sensitive information workflow, etc.
- Web UI Testing Agent: mainly test access to web UI, if blocked, try to achieve privilege.
- Web UI Tampering Agent: mainly tamper sensitive information through web UI.

- Web UI Monitoring Agent: mainly monitor other users' behavior regarding data query/modification through web UI.
- Web Trace Remove Agent: mainly remove traces (modify/delete web server logs, other sensor logs) for malicious behavior to avoid detection.
- File System Testing Agent: mainly test access to file system, if blocked, try to achieve privilege.
- File System Tampering Agent: mainly tamper data by modifying files, documents in the file system.
- File System Monitoring Agent: mainly monitor other users' behavior, such as creating new files, modifying existing files in the file system.
- File System Trace Remove Agent: mainly remove traces (modify/delete file server logs, other sensor logs) for malicious behavior to avoid detection.
- Database Testing Agent: mainly test access to database system, if blocked, try to achieve privilege.
- Database Tampering Agent: mainly tamper data by modifying (update, delete, create) data/entries in the database system.
- Database Monitoring Agent: mainly monitor other users' behavior, such as creating new entries, modifying/deleting existing entries in the database system.
- Database Trace Remove Agent: mainly remove traces (modify/delete database server logs, other sensor logs) for malicious behavior to avoid detection.
- Network Monitoring Agent: mainly monitor other network traffic to collect sensitive information.
- SDR Agent: mainly protect malware/malicious code (software agents) and help to defend against intrusion prevention/detection system (e.g. Snort) as well as other protection software.

We have also designed the collaboration method (TCP connection, message passing, file sharing) and collaboration rules for the automated agents, which have already been implemented.

Algorithm 3. Automated Agent Simulation Algorithm
Input: thread (automated agent) information, automated agent list, automated agent collaboration rule, state transition probability tables.
Output: collaborate with other automated agents to achieve certain goal according to the collaboration rules.

```
1. Hierarchical_State_Machine (thread);
2. If (result = = failure) {
   2.1 Load (automated_agent_list);
   2.2 Load (collaboration_rule);
   2.3 Send_request_to (collaboration_agent);
   2.4 Activate (collaboration_agent);
   2.5 Deactivate (thread);
   }
3. Else if (result = = success) {
   3.1 While (request_list != NULL) {
            Respond_to (reqeust);
            Activate (request_agent);
            Remove (request);
       }
   3.2 Deactivate (thread);
   }
```

behavior and the collaboration among agents to achieve the mission goal. Before simulation, we have pre-designed a set of automated agents and specified the responsibility for each agent. Also, the collaboration rules have been predetermined to guide the collaboration among automated agents. Insider mission progresses as the automated agents cooperate with each other.

3.6 Detection-evasion techniques

According to the Boeing Red Team, malicious insiders may take advantage of a set of detection evasion strategies to keep the insider mission and their activities stealthy. To make our simulation more realistic and flexible, we support varieties of detection-evasion techniques. Automated agents could be configured to utilize the detection-evasion techniques to avoid detection.

We support totally 7 detection-evasion techniques in our simulator:

- Self-throttling: lower the pace of insider mission execution by increasing the time intervals between insider events.
- Leveraging equivalent event sequence (event renaming/event merging): replace a series of insider mission events with equivalent event sequence. For example, modifying a file in the file system can be achieved in two ways: either editing the file or saving the modified file into the file system, or deleting the original file and creating a new file with the same file name and modified content.
- Event obfuscation: disguise insider mission by obfuscating single events. In other words, make single events unrecognizable to detection sensors and system logs. Possible approaches include encrypting the payload of command line, replacing the keywords in sensitive network traffic, etc.
- Noise injection: inject noise into the insider mission process, such as conducting mission irrelevant events, accessing useless assets, etc.
- Event retiming: change the timestamp of critical insider mission events to help disguise insider mission. One possible approach could be tampering system clock.
- Event reordering: reorder the insider events without dependency relationships to help disguise insider mission.
- Removing traces: if the privilege of modifying system logs can be achieved, automated agents can be configured to remove or modify mission critical event logs.

We have implemented all the detection-evasion techniques listed above and you can configure the simulator to adopt whichever detection-evasion techniques you would like.

3.7 Simulator Configuration Flexibility

To mitigate the influences from organization's deployment and behavior models, the simulator provides flexible organization configurations and behavior models. In addition, collaborations among software agents can also be configured to generate data sets for different mission scenarios. Moreover, various detection-evading strategies can be employed by software agents to accomplish insider mission. With sufficient flexibilities

provided, we are pretty confident that the generated insider data sets can partially cover the spectrum of real world insider mission cases.

More specifically, the input and output of the insider mission simulator are shown in Figure 3.3.

1. Asset files

There are totally three asset files in the Asset folder: DB_server.xml, file_system.xml and web_server.xml, each showing the content architecture of one system. They describes the hierarchy information within each system, based on which we randomly generate the complete asset files output in the %ROOT%/DG/Output Files.

2. Data hierarchy

Data hierarchy folder mainly contains four files: dimension_list.xml, activity_list.xml, business_process_list.xml and eventtype_list.xml. These four files form up the abstract hierarchy of our mission scenario, based on which we generate both insider mission data and background data. The top-down hierarchy consists of five layers: insider mission, dimension, activity, business process and event type.

3. Role hierarchy

Role hierarchy folder contains three files: role_list.xml, actor_list.xml and account_list.xml. These three files describe the role model and hierarchy information of actors in our simulation. Also, they provide detailed information for employees involved in the data set.

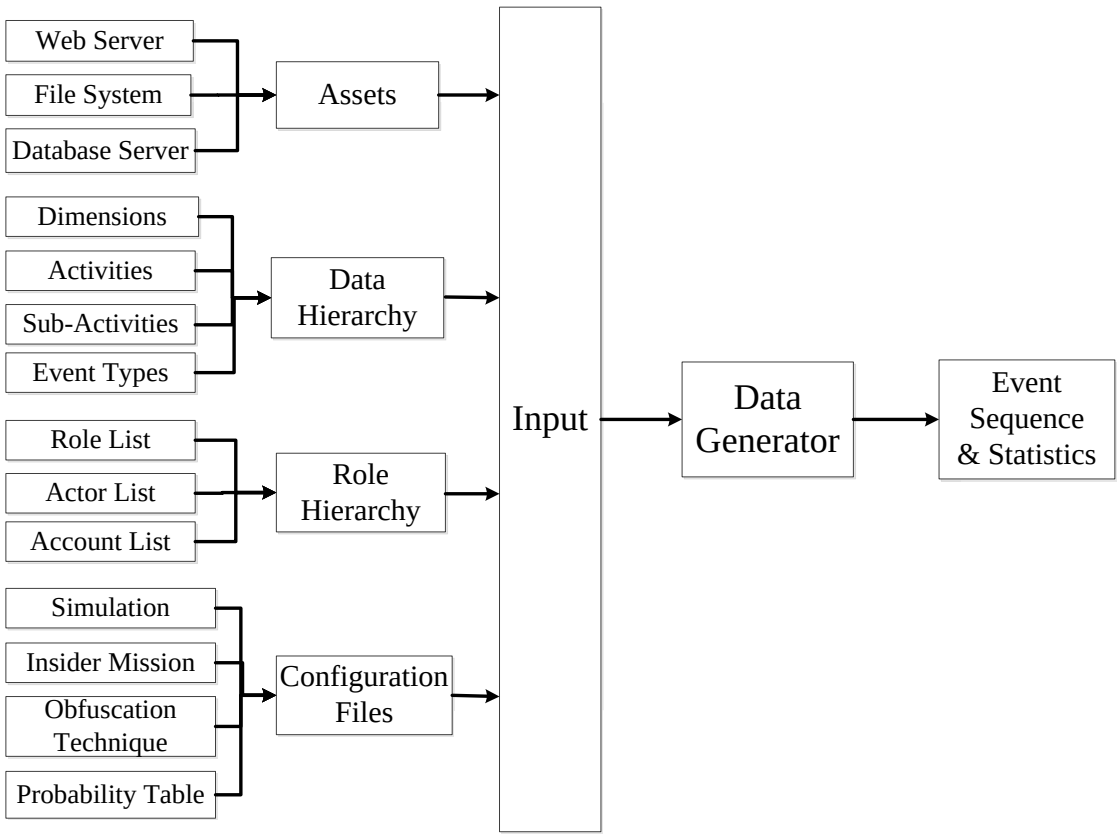

Figure 3.26 I/O of Insider Mission Simulator

4. Configuration files

To improve the flexibility of our simulator, we extract the control parameters out to a set of configuration files. Configuration files folder contains three files: "*Simulation.xml*", "*Insider_Mission.xml*" and "*Obfuscation_Technique.xml*" and a folder named "*Probability Table*":

- *Insider_Mission.xml* file is used to do configurations for insider mission simulation. You can mainly configure four types of information:
  a. Insider mission on/off: whether to generate data set with insider mission or without insider mission. If you want to generate data set with insider mission, you can set the value of "on-off" element as "on"; otherwise, "off".
  b. Start time of insider mission: you can configure when the insider mission starts by setting the value of "start-time" element in the format "mm/dd/yyyy hh:mm:ss".

c. End time of insider mission: you can also configure when the insider mission ends by setting the value of “end-time” element in the format “mm/dd/yyyy hh:mm:ss”.
d. Insider role: what role (security analyst, database administrator or application developer) the insider (hidden behind automated agents) assumes. You can do the configuration by setting the value of “insider-role” element as security analyst, database administrator or application developer.

- *Simulation.xml* file is used to do configurations for the overall simulation. You can mainly configure four types of information:
  a. Random seed: You can configure the random seed for simulation by setting the numerical value of “random-seed” element.
  b. Simulation times: how many times you want to simulate (how many different event lists you want to generate with the same configuration files). You can configure the simulation times by setting the numerical value of “simulation-times” element.
  c. Start time of simulation: you can configure when the simulation starts by setting the value of “start-time” element in the format “mm/dd/yyyy hh:mm:ss”.
  d. End time of simulation: you can also configure when the simulation ends by setting the value of “end-time” element in the format “mm/dd/yyyy hh:mm:ss”.
- *Obfuscation_Technique.xml* file is used to do configurations for obfuscation techniques employed by the automated agents in the insider mission. Here we provide 7 types of obfuscation techniques: noise injection, event retiming, self-throttling, event-reordering, leveraging equivalent event sequence, event obfuscation and removing traces. You can choose which techniques the automated agents use in the insider mission.
  - To include one obfuscation technique, you can set the value of “on-off” element of the technique as “on”. Otherwise, you can set the value as “off”. Also, you can include either none or multiple obfuscation techniques provided.
- *Probability Table* folder contains three files: *Analyst_Status_Trasition.xml*, *DBA_Status_Transition.xml* and *Developer_Status_Transition.xml*. You can configure the behavior model for each role by changing the probability of status transition between job responsibility and daily behavior.

Chapter 4 Evaluation

4.1 Example data sets

We are able to generate data sets of various features specified by different sets of configuration files. Simulators can be configured to include different behavior models, simulation contexts and detection evasion strategies. By default, the simulation time period is from Jan. 1, 2018 to Jan. 30, 2018. With default configuration, each data set contains about 65,000 events.

To validate the basic features of our simulator, we generate eight data sets as examples for analysis. Among these data sets, we have data sets either including insider mission or not including insider mission, data sets with different mission starting time and mission ending time, data sets with different obfuscation techniques, etc.

Figure 4.1 shows an example event instance generated by our simulator:

```
<Event id="10"
 type-id="V0200"
 description="successfully login the TAS web UI"
 activity="D2"
 account="web1025"
 OS-account="a02005"
 user-id="a02005"
 asset-id="1.7.1"
 insider-event="no"
 obfuscation="no"
 start-time="2012/01/01 07:44:41"
 end-time="2012/01/01 07:44:41"
 source-ip="10.28.46.48"
 dest-ip="10.16.48.62"
 dest-port="22" />
```

Figure 4.1 An example event instance

This sample event instance has totally 15 attributes:

- *Event id* is unique to each event instance, which helps to differentiate event instances.

- *Type id* is unique to each event type, which indicates which event type this event instance is.
- *Description* is a brief description of what this event instance does.
- *Activity* shows which activity type this event instance belongs to (this is ground truth information provided for verification purpose, it will be hidden from detection algorithm).
- *Account* shows this event instance is done by which application account.
- *OS-account* shows this event instance is done by which operating system account (os account is one-to-one mapped to employee id).
- *User-id* is unique to each specific user/employee (this is ground truth information provided for verification purpose, it will be hidden from detection algorithm).
- *Asset-id* is a hierarchical id unique to each specific asset (e.g. a webpage, a file, an entry in database).
- *Insider-event* is a label showing whether this event instance is an insider event or not (this is ground truth information provided for verification purpose, it will be hidden from detection algorithm).
- *Obfuscation* is a label showing whether this event instance uses any obfuscation technique (this is ground truth information provided for verification purpose, it will be hidden from detection algorithm).
- *Start time* indicates when this event instance starts.
- *End time* indicates when this event instance ends.
- *Source-ip* indicates the source IP address for this specific event instance.
- *Dest-ip* indicates the destination IP address for this specific event instance.
- *Dest-port* indicates the destination port for this specific event instance.

Table 4.1 shows the basic statistical results of 8 example data sets.

Table 4.1 Overview: Data Sets with Different Features

Data set No.	Total # of events	Time cost (ms)	Insider mission event ratio	Obfuscated event ratio	Relevant background event ratio*	Irrelevant event ratio**
~without insider 1	63,101	698	0%	0%	63.02%	36.98%
~without insider 2	62,547	642	0%	0%	60.34%	39.66%
~without insider 3	61,931	657	0%	0%	60.31%	39.69%
~without insider 4	69,750	710	0%	0%	60.90%	39.10%
~with insider 1	68,366	823	10.90%	0.54%	53.65%	35.45%
~with insider 2	70,812	713	14.21%	0.81%	51.37%	34.42%
~with insider 3	66,990	690	7.21%	2.79%	58.01%	34.78%
~with insider 4	65,787	725	6.76%	2.54%	56.37%	36.87%

* The ratio of the events produced by normal users to maintain daily activities that can be part of insider mission.

** The ratio of the events produced by normal users to maintain daily activities that are not part of insider mission.

4.2 Validation of simulation results

4.2.1 Validation of simulated insider mission instances

In this section, we are going to validate the simulated insider mission instances by answering a set of questions as follows.

Question 1: When we intend to simulate an insider mission instance, does the simulated insider mission instance progress and achieve the mission as expected?

Answer: We configured our simulator and activated the insider mission. The simulation time period is set from Jan. 1, 2018 to Feb. 28, 2018. We configured the insider mission to be activated from Jan. 6, 2018. We executed our simulator for 5 times and generated 5 insider mission data sets. Since we have all the ground truth knowledge of the simulated insider mission instances, we manually went through the 5 insider mission data sets and got the following findings.

Table 4.2 Insider Mission Instances Progress Timeline

Stages / Data set No.	Reconnaissance Stage	Tampering Stage	Cover-up Stage
1	6th day	10th day	18th day
2	6th day	12th day	20th day
3	6th day	10th day	16th day
4	6th day	11th day	19th day
5	6th day	12st day	21st day

We can see that all the insider mission instances are activated on the 6th day. After spending 4 to 6 days in Reconnaissance (Dimension A), all the insider mission instances

entered into the Tampering stage (Dimension B, C and D). After being activated for 10 to 15 days, 5 simulated insider mission instances all completed the Reconnaissance stage, Tampering stage and went to the Cover-up stage (Dimension E and F). After the insider mission being activated for 20 days, all the insider mission instances went through the Reconnaissance stage, Tampering stage and Cover-up stage and then stayed at the Cover-up stage. On the other hand, we manually checked the simulation logs and found that the insider mission specific data in the database and file system had been modified.

In conclusion, the simulated insider mission instances went through the Reconnaissance stage, Tampering stage, Cover-up stage and finally achieved the insider mission as expected.

Question 2: When we intend to simulate an insider mission instance, does the simulated insider mission instance cover all the intended dimensions and business processes?

Answer: We use the same 5 insider mission data sets generated in Question 1 to do the validation for this question. Since we have all the ground truth knowledge of the simulated insider mission instances, we manually went through the 5 insider mission data sets and found that the simulated insider mission instances do cover all the intended dimensions (Dimension A, B, C, D, E and F) and the insider mission business processes.

Question 3: Are the insider mission business processes running as expected? Is the control dependency enforced successfully?

Answer: We use the same 5 insider mission data sets generated in Question 1 to do the validation for this question.

To answer this question, we wrote a python script to do the validation. Firstly, it extracts all the insider mission events from the data sets. Secondly, it checks the extracted event list against the designed insider mission business processes. We did the validation on the 5 insider mission data sets. No control dependency violation is found.

In conclusion, the insider mission business processes are running as expected and the control dependency is enforced successfully.

Question 4: Is the data dependency successfully enforced in the simulated insider mission instances?

Answer: We configured our simulator and activated the insider mission. The simulation time period is set from Jan. 1, 2018 to Feb. 28, 2018. We configured the insider mission to be activated from Jan. 6, 2018.

We added an additional monitoring thread to watch out for each data access event. This additional monitoring thread helps us to check the data dependency while the simulation is running and logging all the abnormal data access events. If there is any data access event violating the data dependency rule, it will output the event and Data ID in the logs. We executed our simulator with this monitoring thread for 5 times and no data dependency violation is found.

In conclusion, the data dependency is successfully enforced in the simulated insider mission simulations.

Question 5: Are all the detection evasion strategies implemented successfully?

Answer: In our simulation, the simulator can be configured to support 7 detection-evasion strategies: self-throttling, leveraging equivalent event sequence, event obfuscation, noise injection, event retiming, event reordering and removing traces.

In order to validate whether these detection-evasion strategies have been implemented successfully in our simulation, we configured the simulator to activate all the detection-evasion strategies and generated 5 data sets with malicious insiders. We wrote a python script to check the 5 data sets and got the following results:

Table 4.3 Number of Events Involved in Different Detection Evasion Strategies

Data set No. / Number of events involved	1	2	3	4	5
Total number of events	127,458	132,101	133,385	129,735	132,518
Self-throttling	135	162	168	152	164
Leveraging equivalent event sequence	326	386	382	345	369
Event obfuscation	187	223	231	204	221
Noise injection	2,256	2,432	2,512	2,366	2,496
Event retiming	126	144	152	136	148
Event reordering	133	154	163	142	156
Removing traces	28	36	38	32	36

Table 4.3 presents the number of events involved in different detection evasion strategies in the 5 data sets. We can see that all the detection evasion strategies have been implemented and presented in the insider mission data sets. In addition, we checked some example detection evasion strategies manually. They all work as expected.

In conclusion, all the detection evasion strategies implemented successfully in the insider mission simulations.

4.2.2 Validation of simulated non-insider-mission business processes and events

In this section, we are going to validate the simulated non-insider-mission business processes and events by answering a set of questions as follows.

Question 1: When we intend to conduct a simulation without insider mission, does the simulation cover the intended business processes as expected?

Answer: We configured our simulator and deactivated the insider mission. The simulation time period is set from Jan. 1, 2018 to Feb. 28, 2018. We executed our simulator for 5 times and generated 5 data sets without insider mission.

Since we have all the ground truth knowledge, we manually went through the 5 data sets without insider mission and found:

1) the 5 data sets do not contain any insider mission specific business processes;

2) the 5 data sets do cover the operational business processes and supporting business processes as expected.

Question 2: When we intend to simulate a non-insider-mission business process, does the simulation result in the expected set of events? Is the control dependency enforced successfully?

Answer: We use the same 5 data sets without insider mission generated in Question 1 to do the validation for this question.

To answer this question, we wrote a python script to do the validation. Firstly, it extracts each business process from the data sets according to the ground truth knowledge. Secondly, it checks the extracted event list against the designed operational business processes and supporting business processes. We did the validation on the 5 data sets without insider mission. No control dependency violation is found.

In conclusion, the non-insider-mission business processes are running as expected and the control dependency is enforced successfully.

Question 3: Is the data dependency successfully enforced in the non-insider-mission simulation?

Answer: We configured our simulator and deactivated the insider mission. The simulation time period is set from Jan. 1, 2018 to Feb. 28, 2018.

We added an additional monitoring thread to watch out for each data access event. This additional monitoring thread helps us to check the data dependency while the simulation is running and logging all the abnormal data access events. If there is any data access event violating the data dependency rule, it will output the event and Data ID in the logs. We executed our simulator with this monitoring thread for 5 times and no data dependency violation is found.

In conclusion, the data dependency is successfully enforced in the non-insider-mission simulations.

4.2.3 Validation of the interleaving between insider mission and non-insider-mission events

Similar to previous sections, for this section, we are going to validate the interleaving between insider mission and non-insider-mission business processes and events by answering a set of questions as follows.

Question 1: When we intend to simulate an insider mission instance, do the interleaving between insider mission and non-insider-mission business processes play a special role in the insider mission instance?

Answer: We use the same 5 insider mission data sets generated in section 4.2.1 Question 1, Since we have all the ground truth knowledge of the simulated insider mission instances, we manually went through the 5 insider mission data sets and got the following findings:

1) these business processes play an important role in progressing and achieving the insider mission;
2) these business processes are also crucial in non-insider-mission daily activities.

Without ground truth knowledge, we have no idea what role these business processes are playing.

Question 2: When we intend to simulate an interleaving business process between insider mission and non-insider-mission business processes, does the simulation result in the expected set of events? Is the control dependency enforced successfully?

Answer: We use the same 5 insider mission data sets generated in Question 1 to do the validation for this question.

To answer this question, we wrote a python script to do the validation. Firstly, it extracts each business process from the data sets according to the ground truth knowledge. Secondly, it checks the extracted event list against the designed operational business processes. We did the validation on the 5 data sets. No control dependency violation is found.

In conclusion, the interleaving business processes are running as expected and the control dependency is enforced successfully.

Question 3: Is the data dependency successfully enforced in the interleaving business process?

Answer: We configured our simulator and activated the insider mission. The simulation time period is set from Jan. 1, 2018 to Feb. 28, 2018.

We added an additional monitoring thread to watch out for each data access event. This additional monitoring thread helps us to check the data dependency while the simulation is running and logging all the abnormal data access events. If there is any data access event violating the data dependency rule, it will output the event and Data ID in the logs. We executed our simulator with this monitoring thread for 5 times and no data dependency violation is found.

In conclusion, the data dependency is successfully enforced in the interleaving business processes.

4.3 Efficiency performance evaluation

We conduct an experiment comparing the efficiency of multi-threaded simulation and that of event-driven simulation. We generate five data sets with insider mission and five without insider mission with multi-threaded simulation method. On the other hand, we generate five data sets with insider mission and five without insider mission with event-driven simulation method. With twenty data sets in total, we calculate the time cost per 10,000 events of simulation with and without insider mission using multi-threaded simulation method and the time cost per 10,000 events of simulation with and without insider mission using event-driven simulation method. The result is presented in Table 4.4.

Table 4.4 Simulation Time Cost: Multi-threaded vs. Event-driven

Simulation time (ms/10,000	Multi-threaded Simulation	Event-driven Simulation
Without insider mission	17163	123
With insider mission	22414	111

- The data is based on measurement of 5 data sets each.
- We only measure the simulation time, not including output time.

From Table 4.4, we can see that event-driven simulation approach is much more efficient than multi-threaded simulation. In addition, as the simulation scales, the time cost of event-driven simulation increases linearly while the time cost of multi-threaded simulation increases much faster. As we shift to event-driven approach, we have improved the data generation efficiency significantly.

The underlying reason is that, in our simulation, we only need event-level concurrency; however, multi-threaded simulation provides instruction-level concurrency. In simulation, one event consists of hundreds of instructions. Frequent switches between different threads consume a great amount of CPU resource, lowering the simulation speed. On the other hand, we have some inherent constraints in simulation (e.g. insider mission

simulation is time sequential); however, operating system has no idea of these constraints. Therefore, operating system is not able to serve as a good scheduler. Based on my observation, operating system often does ad hoc scheduling during simulation, leading to unnecessary collisions and switches among threads. In contrast, the event-driven approach combined with the specially designed scheduler fulfills the requirement of our simulation and reduces unnecessary overhead, thus improving the efficiency greatly.

Chapter 5 Future Work

Currently, the event types are defined by us based on the real audit logs and observables. In the future, we would like to automatically extract event types from real world audit logs. It mainly has three advantages over manual definition: 1) it will improve the efficiency of the definition process; 2) it will provide a more complete set of event types; 3) it will help to avoid biases from human beings.

In addition, the current organizational behavior model is simplistic, which cannot represent the real-world situation very well. Therefore, to generate more realistic insider data, we plan to investigate organizational behavior more and build a model close to the real world. Also, one big limitation is that without real insider data, we do not have an effective way to validate the fidelity of simulated data.

Furthermore, most of the detection evasion strategies implemented prove to be ineffective. To learn more about the strength and weakness of each evasion technique, we plan to investigate more on them and conduct comprehensive studies of the detection evasion techniques.

Chapter 6 Conclusion

In this paper, we have introduced the background of insider threat and narrowed down our research question to achieving flexible and realistic insider mission simulation. Basically, there are two major challenges in getting insider data: 1) Real insider mission data is too sensitive to share; 2) Not all ground truth is known/flagged in real data. To address these challenges, we simulate the insider mission and generate insider data sets by ourselves. More specifically, our approach is to: first, define the event types and business processes based on real world sensors and observables; second, construct the 4-layer insider mission hierarchical state machine based on the insider mission scenarios provided by red team; third, figure out typical user roles and job responsibilities in the organization and construct the behavior models accordingly; fourth, simulate insider mission events as well as background events concurrently and in a time sequential way; finally, output the whole event sequence in XML format after simulation.

We adopt event-driven simulation approach, which proves to be very efficient. To help envision the future trend of insider threat, our simulator supports automated-agent-centric insider mission simulation. 15 automated agents as well as the collaboration rules are designed to conduct the insider mission collaboratively. In order to make our simulation more realistic, we allow insiders or automated agents to utilize varieties of detection-evasion techniques to avoid detection. In addition, to mitigate the influences from organization's deployment and behavior models, our simulator provides flexible organization configurations and behavior models. Moreover, collaborations between software agents can also be configured to generate data sets for different mission scenarios. With sufficient flexibilities provided, we are pretty confident that the generated insider data sets can partially cover the spectrum of real world insider mission cases.

Although simulated data may be biased and different from real world cases, the generated data set has its advantages: 1) we can provide or flag all the ground truth in the data set; 2) we can configure the system parameters to provide different insider mission scenarios and insider behavior patterns. While we admit the shortcomings and limitations of simulated data, it does contribute to the insider threat research.

Bibliography

[1] E. Rich, I.J. Martinez-Moyano, S. Conrad, D.M. Cappelli, A.p. Moore, T.J. Shimeall, D.F. Andersen, J.J. Gonzalez, R.J. Ellison, H.F. Lipson, D.A. Mundie, J.M. Sarriegui, A. Sawicka, T.R. Stewart, J.M. Torres, E.A. Weaver and J. Wiik. Simulating Insider Cyber-Threat Risks: A Model-Based Case and a Case-Based Model. Proceedings of the 23rd International Conference of the System dynamics Society. Boston, MA, 2005.

[2] R. F. Mills, G. L. Peterson and M. R. Grimaila, 2009. Insider Threat Prevention, Detection and Mitigation. In Cyber-Security and Global Information Assurance: Threat Analysis and Response Solutions, K. Knapp, editor, Hershey, PA: IGI Global Publishing.

[3] CERT. (2007). 2007 E-Crime Watch Survey. Retrieved from: http://www.cert.org.

[4] D. Capelli, A. Moore, R. Trzeciak, T. Shimeall, 2009. Common Sense Guide to Prevention and Detection of Insider Threats. Carnegie Mellon University Cylab. version 3.1

[5] M. R. Randazzo, M. Keeney, E. Kowalski, D. Cappelli, and A. Moore. Insider Threat Study: Illicit cyber activity in the banking and finance sector. Technical report, U.S. Secret Service and CERT Coordination Center/SEI, 2004.

[6] K. S. Killourhy and R. A. Maxion. Toward Realistic and Artifact-Free Insider-Threat Data. 23rd Annual Computer Security Applications Conference, pages 87-96, 2007. ACSAC 2007.

[7] S. Greenberg. Using Unix: Collected traces of 168 users. Technical Report 88/333/45, Department of Computer Science, University of Calgary, Calgary, Canada, 1988.

[8] R. A. Maxion. Masquerade detection using enriched command lines. In International Conference on Dependable Systems and Networks (DSN-03), pages 5–14. Held on 22–25 June, 2003, San Francisco, CA, IEEE Computer Society Press, Los Alamitos, CA, 2003.

[9] T. Lane and C. E. Brodley. An application of machine learning to anomaly detection. In Proceedings of the 20th Annual National Information Systems Security Conference, pages 366–380. Held on 7–10, October, 1997, Baltimore, MD, NIST, 1997.

[10] M. Schonlau, W. DuMouchel, W.-H. Ju, A. F. Karr, M. Theus, and Y. Vardi. Computer intrusion: Detecting masquerades. Statistical Science, 16(1):58–74, 2001.

[11] Malek Ben Salem, Shlomo Hershkop, Salvatore J. Stolfo. \A Survey of Insider Attack Detection Research" in Insider Attack and Cyber Security: Beyond the Hacker, Springer, 2008.

[12] R. Chinchani, A. Muthukrishnan, M. Chandrasekaran, and S. Upadhyaya. RACOON: Rapidly generating user 17command data for anomaly detection from

customizable templates. Computer Security Applications Conference, Annual, 0:189-204 (2004)

[13] A. Garg, V. Sankaranarayanan, S. J. Upadhyaya, K. A. Kwiat. USim: A User Behavior Simulation Framework for Training and Testing IDSes in GUI Based Systems. Annual Simulation Symposium, 196-203 (2006)