A Model for Calculating Damage Potential in Computer Systems

by

Sharad Nolastname

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved December 2018 by the
Graduate Supervisory Committee:

Rida Bazzi, Chair
Arunabha Sen
Adam Doupé

ARIZONA STATE UNIVERSITY

May 2019

ABSTRACT

For systems having computers as a significant component, it becomes a critical task to identify the potential threats that the users of the system can present, while being both inside and outside the system. One of the most important factors that differentiate an insider from an outsider is the fact that the insider being a part of the system, owns privileges that enable him/her access to the resources and processes of the system through valid capabilities. An insider with malicious intent can potentially be more damaging compared to outsiders. The above differences help to understand the notion and scope of an insider.

The significant loss to organizations due to the failure to detect and mitigate the insider threat has resulted in an increased interest in insider threat detection. The well-studied effective techniques proposed for defending against attacks by outsiders have not been proven successful against insider attacks. Although a number of security policies and models to deal with the insider threat have been developed, the approach taken by most organizations is the use of audit logs after the attack has taken place. Such approaches are inspired by academic research proposals to address the problem by tracking activities of the insider in the system. Although tracking and logging are important, it is argued that they are not sufficient. Thus, the necessity to predict the potential damage of an insider is considered to help build a stronger evaluation and mitigation strategy for the insider attack. In this thesis, the question that seeks to be answered is the following: 'Considering the relationships that exist between the insiders and their role, their access to the resources and the resource set, what is the potential damage that an insider can cause?'

A general system model is introduced that can capture general insider attacks including those documented by Computer Emergency Response Team (CERT) for the Software Engineering Institute (SEI). Further, initial formulations of the damage

potential for leakage and availability in the model is introduced. The model usefulness is shown by expressing 14 of actual attacks in the model and show how for each case the attack could have been mitigated.

To Megha, Saurav and Monika Bharti

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my thesis advisor, Professor Rida Bazzi, for his continuous motivation, patience, and support throughout the research. I express my sincere gratitude to him for giving me the intellectual freedom in my work while demanding a high quality of effort in all my endeavors.

Besides my advisor, I would like to thank my thesis committee members, Prof. Arunabha Sen and Prof. Adam Doupé for their insightful comments and interest in my work.

I thank all my friends at the university for their continuous encouragement and stimulating discussions. Finally, I would like to thank my parents, brother, sister, and life partner for their love and unyielding support during the different stages of my research.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF FIGURES

Chapter 1

INTRODUCTION

There has been a significant rise in risks, threats, and several forms of attacks on the organizations, where information plays a crucial role (PwC 2015). For systems having computers as a significant component, it becomes a critical task to identify the potential threats that the users of the system can present, while being both inside and outside the system. Most of the initial works done in the field of security identify themselves with the outsider threat, proposing models and techniques, such as attack graphs (Phillips, Swiler et al. 1998; Sheyner, Haines, et al. 2002; Swiler, Phillips et al. 2001) Intrusion Detection Systems (Porras and Neumann 1997; Stanifor-Chen, Cheung et al. 1996), and Access Control Lists (Zhang, Li, et al. 2004). But a growing reliance of organizations on information infrastructures and individuals who own the capabilities for such infrastructures made the organization increasingly vulnerable to threats from the insiders. This demanded a shift in research paradigm towards insider threat.

One of the most important factors that differentiate an insider from an outsider is the fact that the insider being a part of the system, owns privileges that enable him/her access to the resources and processes of the system through valid capabilities (Bishop and Gates 2008). Another aspect of this difference is the amount of knowledge that an insider has about the organization's system. Such knowledge can include not just the information about resources, but also the location of the resources along with the capabilities of other employees having access to these resources. This knowledge, overall, exposes partial or full internal structure of the system to the insider.

Despite the fact that trusting insiders is paramount to the functionality of an

organization, a full and blind trust could create a potential for misuse of their capabilities. An insider with malicious intent can potentially be more damaging compared to outsiders. The above differences help us understand the notion and scope of an insider.

The implication of attack from the insiders is clearly manifested by the findings of Moore et al. (2008), according to which the monetary losses, due to insider attacks, ranged from five hundred dollars to tens of million dollars. Around 75 percent of the organizations had a negative impact on their business operations and 28 percent experienced a negative impact on their reputation as an affect of insider attack (Baracaldo and Joshi. 2013). The report by U.S State of Cybercrime Survey (PwC 2015), sponsored by CERT insider threat center, found that 23 percent of electronic crime events were suspected or known to be caused by an insider. In the 2017 survey, by the same division of CERT (PwC 2017), nearly 30 percent of all respondents reported that incidents caused by insider attacks were more costly or damaging than outsider attacks. Among the attacks where insiders were the source, 35 percent of incidents had private or sensitive information intentionally exposed by them.

The significant loss to organizations due to the failure to detect and mitigate the insider threat has resulted in an increased interest in insider threat detection. The well-studied effective techniques proposed for defending against attacks by outsiders have not been proven successful against insider attacks (Colwill and Carl 2009). Although this has led to the development of different security policies and models which directly deal with the insider threat, the approach taken by most organizations is the use of audit logs such as the one suggested by Anderson and Brackney (2004), after the attack has taken place. Such approaches are inspired by academic research proposals to attack the problem by tracking activities of the insider in the system. Although tracking and logging are important, we argue that they are not sufficient.

With the in-depth knowledge of the system, malicious insiders tend to create an attack path which can remain invisible to such tracking methods. Thus, we consider that a method of predicting the potential damage based on the knowledge of the insider which the insider accumulates from his/her access to the resources in the system can help build a stronger evaluation and mitigation strategy for the insider attack.

In this thesis, the question we seek to answer is, 'Considering the relationships that exist between the insiders and their role, their access to the resources and the resource set, what is the potential damage that an insider can cause based on future events from use of such accesses?'

1.1 Approach

Security agencies and organizations such as CERT have successfully documented several instances of insider attacks in the form of use cases and determined the patterns and trends of malicious activities (Silowash, George et al. 2012). Detailed classifications of incidents have been provided based on where the threats can be detected within the target system. We start by utilizing these use cases and classifications to formulate an exhaustive model in which various scenarios discussed in the use cases can be expressed.

Such a model is not straightforward. In fact, our study of the CERT cases shows that the insiders can cause damage either by remaining within their default set of privileges or exceeding it by seeking new capabilities. In the latter case, the exposed value to the insider could increase with the augmentation of new capabilities. We want our model to represent access to and propagation of the information. For example, "it is one thing to control access to a file and another to control access to all information in the system derived from or dependent upon the data in the file" (Cohen, Ellis, et al. 1975). Thus, with an established model, we take into consideration, various

aspects of data leakage and availability compromise through capability augmentation and see how it can contribute to the calculation of the final exposed value.

1.2 Contributions

The contributions in this thesis are structured around the central questions of how to calculate the damage potential of an insider working for an organization having access to the computer system. Our model is motivated by Hydra, an object-based computer system with capability developed for the creation of a highly secured system and by the elements of graph theory. The unique contributions of this thesis are listed below.

- Differing from the previous systems that suggest the use of attack paths (Chinchani, Iyer. et al. 2005) and those which uses the risk values derived from the trust factors (Baracaldo and Joshi. 2013), to calculate the risk of occurrence of threats , we try to focus on how a legitimate user can access the system around him and use his capabilities to affect the state of the system. In this work, we analyze the nature of information in the computer system and the insider's capabilities to move the information within or outside the system.

- As our next contribution, we model the capability of the insider to affect the availability of the system. To our current knowledge, no models which tried to predict the threat of the insider considered the availability aspect of the system. Our study of CERT insider attack cases reveals that among the 50 cases analyzed, 20 had the insider affecting or trying to affect the availability of the system.

- Another important contribution of this work is the creation of a corpus of selected cases, where we analyze each type of data that has been accessed, the

access rights that were available to the insider, and the channels used for the attack. Further, we provide the representation of the cases using our model which would help us understand the ways that malicious insiders use to launch the attack in the system. We also provide mitigation strategies based on threat actions that explain how our model could be used to prevent similar attacks.

1.3 Organization

The remainder of the thesis is organized as follows. Related work is discussed in Chapter 2. In Chapter 3, we describe our proposed model and define various components that build a computer system. Information Flow and the damage potential associated with it are given in Chapter 4. The Availability Damage is described in Chapter 5. In Chapter 6, we apply our model to several use cases. Chapter 7 concludes the thesis.

Chapter 2

RELATED WORK

The research community has focused on various modes of insider threat detection and prevention such as attack paths, attack strategies, intrusion model, access control and risk-assessment processes. But the analysis and calculation of damage potential of an employee within the organization have not yet received the appropriate research focus. Silowash suggest that the organizations must identify those high-risk users who most often interact with the critical systems or data (Silowash, George, et al. 2012). Such an identification is a complex task and requires an elaborate study of the system and its user's capabilities. Various approaches have been suggested in order to identify such users and resources in order to mitigate insider threat.

One of the few recent works on the estimation of threat by an insider belongs to Chinchani et al. The authors used a threat model called key challenge graph which focuses on the legitimate user's view of an organization network (Chinchani et al. 2005). It relies on the representation of capabilities present as any vertex as a key and the presence of a security measure or policy that protects the resources as a key challenge on the channel of communication. It then identifies the scope of threat by defining the cost of an attack that the insider can perform. Although we follow a similar graph representation for our model, in our approach, we focus on the damage that an insider can cause as opposed to the cost of individual attacks as proposed by them. A Key Challenge Graph has also been used by Ha et al who introduced an information-centric and graph-oriented tool which takes the physical network topology and a predefined set of vulnerabilities as external inputs, combined with the network translational tools and cost analysis for constructing a key-challenge

graph (Ha, Upadhyaya, et al. 2007). Unlike our work, which looks at the graph from the perspective of the user's capabilities, they used the graph to know the likely attack strategies, vulnerable points, and critical points of the attackers based on the cost of attacks.

In the context of our work, other relevant proposed methods are those based on the use of attack graphs and privilege graphs. The attack graph is usually used to determine whether attackers starting from a specific location can gain normally restricted privilege levels on one or more important targets (Lippmann and Ingols 2005; Ammann 2002). Starting from a source, (Artz 2002) proposes to find all paths to all possible goals and perform reachability computation. It then determines the visibility of attack paths to the intrusion detection system. Taking this approach a step further, (Gorodetski 2016) uses formal grammar to specify allowable paths and then generate individual random paths from the grammar. With the final goal of preventing the attacker from reaching the goal state, (Jajodia, Noel, et al. 2003) reads vulnerability and reachability information, computes attack graphs using the approach described in (Ammann 2002) and makes the recommendations to prevent access to critical hosts. A weakness of these approaches is the assumption that reachability information is available prior to computing attack graphs. Determining such information can be computationally complex task (Lippmann and Ingols 2005). Differing from the attack graphs, our approach does not limit itself to exposure of paths and reachability problem but goes beyond to calculate the damage associated with each found path.

A related technical approach taken by some researchers is of risk assessment based on the potential risks of illegal access via role misuse or abuse. (Baracaldo and Joshi 2013) presented a framework that extended the role-based access control (RBAC) model by incorporating a risk assessment process and the trust system has in its users. It explains trust as dependent on the context in which the interaction between

entities takes place and explains risk by the likelihood of a hazardous situation and its consequence if it occurs. It then calculates the risk using the expected value formula (Celikel et al. 2009), where the probability of occurrence is multiplied by the cost of the event. (Salim et al. 2011) uses the costs of access assigned to permissions depending on the risk of their operations with each user being assigned a budget. The activation of a role by a user depends not just upon the assigned roles to him but also on the cost of activating the role which is bounded by the budget. In another effort, (Chen et al. 2001) proposed a model which uses the trustworthiness of a user to calculate the risk associated with a role, and the degree of appropriateness of the permission-role assignments. None of these techniques takes into account how the information accessed by a role can be used malevolently against the interest of the organization, which is the prime focus of our model.

Behavioral theories have also been used for the development of preventive models for insider threats. (Schultz 2002) described a theoretical prediction model based on five behavioral indicators, "deliberate markers, meaningful errors, preparatory behavior, correlated usage patterns and verbal behavior", which could be counted together to predict and detect insider threats. Similarly [Wood 2000] identifies various aspects of users who pose insider threat and claims that users are assumed to have the skills needed to attack the system and to be risk-averse. Some works also present a comprehensive view of psychological approaches to detecting the insider threat. (Greitzer and Frincke 2010) described this approach by identifying factors such as disgruntlement, anger management, disengagement, performance and absenteeism as predictors of insider threat together with the use of Bayesian-Net led predictive model. On the same note, several different theories on the behavior of insiders are presented by (Theoharidou, Marianthi, et al. 2005), among which situational crime prevention theory suggests that cyber crimes occur when a person has both motive and

opportunity- so by either removing motive or denying a malicious user an opportunity, we can help prevent the crime. These approaches are limited because of their methods of predicting the threat revolves around only the behavioral aspect of the threat. We choose to take an approach of not predicting the attack but evaluating the potential attacks, if they ever happen.

Potential leaks and compromises have been calculated using various activity and risk analysis models that are built around the database activities. (Celikel 2009) proposed a probabilistic risk management model for enhanced access control in distributed databases. They incorporated failure modes and effective analysis (FMEA) for measuring user risks in their design. (Harel, Amir, et al. 2012) presented a new concept, Misuseability Weight, for estimating the risk emanating from data exposed to insiders. It assigns a score that represents the sensitivity level of the data exposed to the user and by that predicts the ability of the user to maliciously exploit this data.

As previously stated, none of the above-proposed methods comprehensively consider the capabilities to transfer the data as well as affect the availability of the system together to estimate the potential damage to an organization by any given user.

Chapter 3

MODEL

The proposed model is designed to calculate the damage potential of an insider by considering the insider's access to the various resources in the computer system. In our model, we assume that the insider has some authorization in terms of access rights and can use these rights to perform actions that affect the system. Furthermore, we assume that the insider and the potential target are not necessarily at the same location and thus one or more intermediate channels of interaction are needed to establish communication between the insider and the target. Since the system has a finite set of tasks that can be performed, we assume that any action performed in the system is a combination of such predefined tasks. We consider that the insider has an initial authentication to use the system.

3.1 Definitions

Prior to the formal definition of the system model and the actions that could be performed within the system, we describe the various components.

Definition 3.1 The *object* set O is a set of all physical and logical resources available to be accessed by a process in the system. For example, disks, files, source codes can be represented as objects. An object has the following components: Type, Nature, Data and Capability List (Figure 1).

1. *Type* determines what operation can be performed on the object. It could be of either `Procedure` or `Text`.

Figure 3.1: Computer System's Object

2. *Nature* represents the nature of the resource that the object represents and is assumed to be related to the value of the object in the real world. For instance, some objects require that their integrities be well guarded, other objects are sensitive (their leakage would result in a lot of damage to the organization), while others may be critical and sensitive simultaneously.

3. *Data* part represents the information stored in the object

4. *Capability List* (C-list) represents the capability that an object of type `procedure` has. This field is empty for the objects that are not of type procedure.

Definition 3.1.1 The *Procedure* set Pr is a set of objects of type procedure which can be executed to create new processes. Each procedure has a C-list associated with it. We assume that the set of objects in the system is finite.

Definition 3.2 The *Process* set P is a set of processes executing in the system. This set is not fixed. Processes can be created and killed. A process can create another process by executing a procedure and can pass capabilities to the newly created process. The newly created process inherits the capabilities already present in the procedure along with the capabilities passed to it by the process creating it. In order for a process to execute a procedure, it should own `exec` capability for the procedure.

Definition 3.3 The *vertex* set V is a finite set of physical nodes where processes

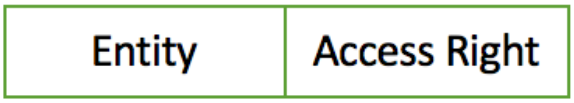

Figure 3.2: Computer System's Capability

and objects can reside.

Definition 3.4 The *channel* set represents the set of channels in the system. Each channel is defined by two vertices, possibly the same. Objects can be transferred across channels and processes can be executed or created across channels.

$$channel \subseteq V \times V$$

Definition 3.5 The *capability* set C is a set of tokens which gives the process a permission to perform an action on an object or a process in the computer system. It is represented as a tuple containing a unique *entity* and the access right (figure 2), where the *entity* can be either an object or a process. The access right defines the *operation* that can be performed on the entity and can be, but is not limited to, *read*, *write*, *execute*, or *kill*. For example, $(antivirus, kill)$ is a capability describing that a process which owns it can kill the antivirus process.

Definition 3.6 (Capability of a process)

$Cap : P \rightarrow \mathcal{P}(O \times A)$ is a function which determines the set of capabilities possessed by a process $p \in P$. For example, $Cap(p) = [(o, read), (o', copy)]$ means that process p has two capabilities: *read* of o and *copy* of o'.

Definition 3.7 (Resource at a vertex)

$Res : V \rightarrow \mathcal{P}(O \cup P)$ is the resource mapping from a node to the set of objects

and processes that it contains. Thus, the set of processes and objects that can also be expressed as:

$$P = \bigcup_{v \in V} Res(v) \setminus O$$
$$O = \bigcup_{v \in V} Res(v) \setminus P$$

Definition 3.8 (Location of a resource)

$Loc : P \cup O \to V$ is the location mapping from processes or objects to their respective location, i.e. the vertex at which they reside.

Definition 3.9 (Value of a resource)

$val : P \cup O \to \mathbb{R}_{\geq 0}$ is a mapping from entities to their value in the system. The value of all entities is positive.

Definition 3.10 (Adjacent Vertices)

$Adj : V \to \mathcal{P}(V)$ is a function which maps a given node v to a set of adjacent nodes. Adjacent nodes are the nodes which are directly reachable from the given node.

Definition 3.11 (System State) Given a system with set of vertices and adjacency relation on these vertices, the system state specifies the state of each vertex in the system and the capabilities of all processes in the system. The state of a vertex includes the resources that reside at the vertex. Formally, specifying the system state consists of two mappings that specifies where objects and processes are and what capabilities they have.

- Resources: specifies for each vertex the resources at the vertex. $Res : V \mapsto \mathcal{P}(O \cup P)$.

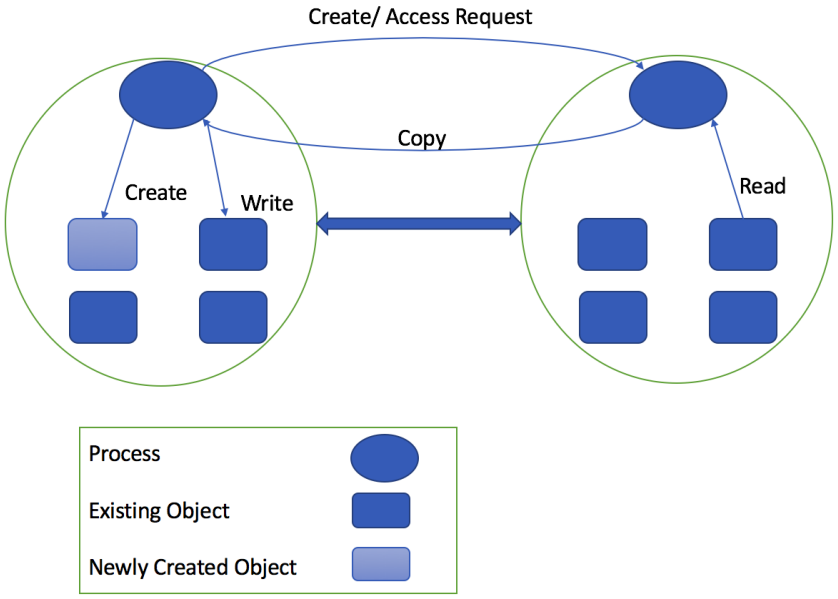

Figure 3.3: Computer System Overview

- Capabilities specifies for each process in the system the capability of the process: $Cap : P \mapsto \mathcal{P}(O \times A)$.

The state of the system is then defined as a

$$state = \langle Res \times Cap \rangle$$

. Let *State* be the set of all possible states in the system.

3.2 Creation of Process at a Node

In the system with set of vertices V, set of processes P, and a set of capabilities C, a process $p \in P$ running at node v can create a process p' at v' which could be the same node as v or a node adjacent to v, by executing the procedure o which transitions the system to a state where p' gets added to the existing set of resources at v'. Formally, the action $createProcess(\langle Res, Cap \rangle, v, v', p, c)$ is enabled if

- $o \in Pr$,
- $Cap(p) \ni (execute, o)$,
- $v = Loc(p)$,
- $v' = Loc(o)$, and
- $v' \in adj(v) \vee v' = v$, then

After executing the action, we have the new system state $\langle Res', Cap' \rangle$ where $Res'(v') = Res(v') \cup p'$ and $C = C \cup Cap(p')$ and $Loc(p') = v'$ and $Res'(x) = Res(x)$ for $x \neq v'$.

3.3 Access of Objects

A process can delete an object residing at the same node and it can read and write objects residing at the same node or at an adjacent node if the process has the required capability for the action.

Deletion of an object In the system with set of vertices V, set of processes P, and a set of capabilities C, the action $a =' delete'$ can be performed on an object $\{o\} \in O$ at node v by process p running on the same node v. It transitions the system in which all resources are unchanged except at node v for which the object is deleted from the set of resources. The deletion of an object is enabled if:

- $v = Loc(p)$
- $v = Loc(o)$
- $Cap(p) \ni (delete, o)$

A function $delete : State \times Vertices \times Processes \times Capabilities \longrightarrow State$ is defined by

$$delete(\langle Res, Cap \rangle, v, p, c) = \langle Res', Cap \rangle$$

where $Res'(v) = Res(v) - o$, $Res'(x) = Res(x)$ for $x \neq v$.

The effect of a process accessing an object for reading or writing is not explicitly modeled in the system state. Such action transfer data between processes and objects and we reason about their effect when we model the damage potential by considering possible executions in the system.

Reading of object In a system with set of vertices V, a set of processes P, and a set of capabilities defined by Cap, the *read* action can be performed on an object $\{o\} \in O$ at node v' by process p running on node v, where v and v' are the same node or are adjacent nodes. The read action is enabled if

- $v = Loc(p)$
- $v' = Loc(o)$
- $v' \in adj(v)$ or $v' = v$
- $Cap(p) = (read, o)$

Writing of object In a system with set of vertices V, a set of processes P, and a set of capabilities defined by Cap, the *write* action can be performed on an object $\{o\} \in O$ at node v' by process p running on node v, where v and v' are the same node or are adjacent nodes. The write action is enabled if

- $v = Loc(p)$
- $v' = Loc(o)$
- $v' \in adj(v)$ or $v' = v$
- $Cap(p) = (write, o)$

3.4 Killing of Process at a Node

A process can kill another process at the same or an adjacent node if it has the required capability. The *kill* action is enabled if:

- $v = Loc(p)$
- $v' = Loc(p')$
- $v' \in adj(v)$ or $v' = v$
- $Cap(p) \ni (p', kill)$

A function $killProcess : State \times Vertices \times Vertices \times Process \times Process \times Capabilities \longrightarrow State$ is defined by:

$$killProcess(\langle Res, Cap\rangle, v, v', p, p', c) = \langle Res', Cap\rangle$$

where $Res'(v') = Res(v') - p'$ and $C' = C \backslash Cap(p')$ In the system with set of vertices V, set of processes P, and a set of capabilities C, a process $p \in P$ running at node v can kill a process p' at v' which can be same as node v or adjacent to v, which transitions the system to a state where p' is removed from the existing set of resources at v'.

3.5 Illustration

An Example showing deletion of an object Consider a set of operations by a process aimed at deleting an object located at an adjacent node. Let $\langle Res, Cap\rangle$ be the initial state of the system (assuming V and the adjacency information is given). The steps that would be required for satisfying the request would be of the form:

$$\langle Res, Cap\rangle \xRightarrow[\text{createProcess(v, v', p, (execute, o))}]{} \langle Res', Cap\rangle \tag{3.1}$$

where $Res'(v') = Res(v') \cup p'$ and $C' = C \cup Cap(p')$

$$\xRightarrow[\text{delete(v, p, (delete, o))}]{} \langle Res'', Cap \rangle \tag{3.2}$$

where $Res''(v) = Res'(v) - o$

$$\xRightarrow[\text{killProcess(v, v', p, (kill, p'))}]{} \langle Res''', Cap' \rangle \tag{3.3}$$

where $Res'''(v') = Res'(v') - p'$ and $Cap' = Cap - Cap(p')$

3.6 Effective Damage Potential

To calculate the effective damage potential of a process, we need to consider damages that could be incurred by both the transfer of data as well as by the affect on the availability of the system. This damage depends on the capabilities initially possessed by a process along with the capabilities gained with the creation of new processes by it. Thus the effective capability of a process is defined as

$$EffectiveCapability(P) = Cap(P) \bigcup_{P' \text{ created by } P} EffectiveCapability(P')$$

Chapter 4

INFORMATION FLOW IN THE COMPUTER SYSTEM

An information flow policy within a computer system specifies a set of nodes and a flow relation defining permissible flows among these nodes.

In order to characterize the admissible transfer rate for the information flow from a single node, let us revisit our definition of computer system network. The computer system is represented by a directed graph $G(V, E)$, where information can be sent noiselessly from v to v' through $(v, v') \in E$, where E represents the set of channels.

Definition 4.1 (Capacity of a channel) The *capacity* of a channel is the maximum permitted rate of object access (in bit per unit time) between two nodes.

$$H_{V \times V} : V \times V \to \mathbb{R}_{\geq 0}$$

A local access within the same node is a special case of object access by the process running on the node. This makes the access relation among the nodes a partial order relation.

Definition 4.2 (Flow) *Flow* $f_{x,y}$ is defined as the rate of transfer of information (per unit time) stored in x to y, where x and y could be an object or a process. An access specifies a flow in one of the following ways

- From an object to a process $o \to p$
- From an object to an object $o \to o'$
- From a process to an object $p \to o$

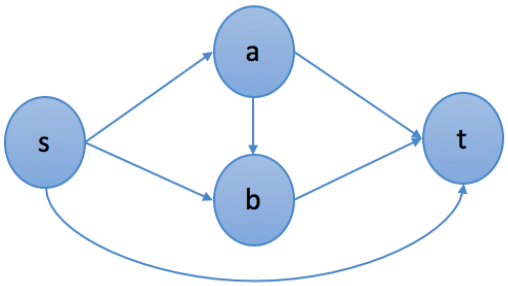

Figure 4.1: Information Flow

- From an object to a process recursively through objects $o \xrightarrow{*} o' \rightarrow p$
- From an object to a process recursively through processes $o \xrightarrow{*} p \rightarrow p'$

A $flow$ is admissible only if the process initiating the flow has the required capability for the transfer of information, and if the sum of all such flow is bounded by the capacity of channel between the vertices on which the source and the destination reside.

$$\sum_{x,y \in (O \cup P) \times P} f_{x,y} \leq H_{v,v'} \text{ where } Loc(x) = v, Loc(y) = v'$$

Definition 4.3 (Value of a Flow) $valF : F \rightarrow \mathbb{R}_{\geq 0}$ is a mapping from flow to their value in the system. It is the value of all the data transferred in the flow. The value of all flows are positive.

4.1 Max-Flow and All-Path Algorithms

4.1.1 Max Flow

To calculate the maximum rate at which information can be transferred from a source node to a target node, we use the Push-Relabel algorithm (Algorithm 1), which

is considered one of the most efficient maximum flow algorithm. Throughout its execution, the algorithm maintains a "preflow" and gradually converts it into a maximum flow by moving flow locally between neighboring nodes using push operations under the guidance of an admissible network maintained by 'relabel' operations.

For example, let the capacities of the channels for the graph shown in figure 5 be as following: $H(s,a) = 2, H(s,b) = 2, H(a,b) = 2, H(a,t) = 2, H(b,t) = 1, H(s,t) = 0$. In this case, we can calculate the value of max flow from s to t is 3.

Let $maxF$ be the maximum flow obtained from the algorithm. Thus, the total value of information that could be transferred from the source vertex to a target vertex is given by:

$$val(maxF)$$

4.1.2 All paths between the source and target

For a critical resource, we need to find all the paths to it to monitor the flow through them. This is accomplished by the recursive algorithm below which finds all the available paths between two nodes s and t in the system. The function $canAccess : V \times V \rightarrow [True, False]$ determines if any process p at a node v can transfer the objects to node v' based on its available capabilities to create a process and access objects at the adjacent node.

Consider the example in Figure 4 which illustrates the available channel between the source and target nodes. If we suppose that node s has capability to access node a, node a has capability to access the node b and t and node b has a capability to access t, then using the above algorithm, the available paths between s and t would be [s, a, t] and [s, a, b, t].

Algorithm 1 To find maximum rate at which objects can be accessed at the destination node

$s \leftarrow$ source *vertex*

$t \leftarrow$ target *vertex*

$V \leftarrow$ list of *vertices*

$E \leftarrow$ list of *Edges*

$G \leftarrow$ Input *Graph*

$G' \leftarrow$ Residual *Graph*

procedure INITIALIZE PREFLOW(N)

Initialize height and flow of every vertex as 0

Set original labelling $d(s) = n, d(v) = 0, \forall v \in V, v \neq s$

Send $c(s - v)$ flow on each edge $s - v \in E$and $v \in V$

For each node v with $s - v \in E, v - s \in G'$ and excess $e(v) = c(s - v)$

procedure PUSH(U)

if $\exists v$ with admissible edge u-v$\in E_f$ **then**

Send $\delta \doteq min(c(u - v), e(u))$ from u to v

$e(u) \leftarrow e(u) - \delta$

$e(v) \leftarrow e(v) + \delta$

procedure RELABEL(U)

for each u-v from u **do**

if $u - v \notin E_f ord(v) \geq d(u)$ **then**

$d(u) = 1 + min_{(v:uv)\in E_f} d(v)$

procedure MAIN LOOP()

while (**do** Push() or Relabel() returns true)

$Push()$ or $Relabel()$

return flow

Algorithm 2 Brute force algorithm to find all the paths with permitted flow of information between two given nodes

$s \leftarrow$ source *vertex*

$t \leftarrow$ target *vertex*

$P \leftarrow$ list of *vertices*

$Path \leftarrow$ list of P

procedure FINDALLPATHS(V),

$P.append(v)$

for `each vertex v' in adj(v)` **do**

if $canAccess(v, v') = true$ **then**

if $v' = t$ **then**

$Path \leftarrow Path \cup P$

else if $adj(v') = NULL$ **then**

return

else if $v' \notin P$ **then**

$FindAllPath(v')$

$P \leftarrow P - P[length\ of\ P]$

return

Chapter 5

AVAILABILITY IN THE COMPUTER SYSTEM

To formally define availability, we need to introduce the concept of *service*.

Definition 5.1 (Service) *Service $s = P_s \cup O_s$ is a set of processes and objects.*

The processes and objects of a service are those processes and objects necessary for providing the functionality that the service implements. A variety of services can be provided by a computer system. For example, rendering of a patient portal page with the patient's information can be a service provided by a health-care organization. Every service requires a set of processes and objects which can be made available to the client of the service.

For a given service $s = P_s \cup O_s$, each object or process in $P_s \cup O_s$ might be available (operational) or not available, but the availability of the service does not necessarily require the availability of all resources of the service. It is possible that some resources are interchangeable. Also, it is not always the case that the availability of a service requires full connectivity among all its entities. For example, Figure 4 shows an application's back-end system. Each server processes have their own database objects along with the shared database object. Both the processes provide simultaneous services to the web request, and the services are said to be available if either of the process and local objects pair is available and the shared object is available. Such conditions are expressed by an *assertion*, which is a predicate that capture the requirements on the availability of individual and groups of resources for the system to be considered available.

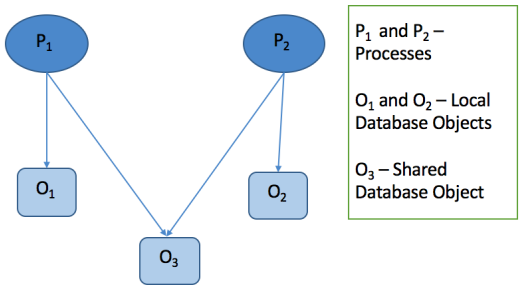

Figure 5.1: Example of a Service

As an example the following assertion for the service in Figure 5.1

$$(P_1 \wedge O_1 \wedge O_3) \vee (P_2 \wedge O_2 \wedge O_3) \tag{5.1}$$

states that if P_1, O_1 and O_3 are available, the system is available. Also if P_2, O_2 and O_3 are available, the system is available.

Definition 5.2 (Availability of a Resource) $avail : P_s \cup O_s \rightarrow \{True, False\}$ is the availability function whose value determines if the process or object in the system is available to be accessed by the client through the services.

Definition 5.3 (Minimum Resources for Availability) $minAvailable : S \rightarrow \mathcal{P}(P_s \cup O_s)$ is the set of all such set of processes and objects whose combination satisfies the assertion for the service's availability. For the example in Figure 4, $\{(p_1, o_1, o_3), (p_2, o_2, o_3)\}$ would be the $minAvailable$ set, with each of its elements satisfying the assertion at (5.1). We can call the service available, if all the entities are available for at least one element of this set.

Thus, the set of process and objects required by a service can be formally written as:

$$O_s = (\bigcup_{e \in minAvailable(s)} e) \cap O$$

$$P_s = (\bigcup_{e \in minAvailable(s)} e) \cap P$$

We further define the service to exhibit one of the following types of availability:

- **Full Availability** A service $s \in S$ in the system is said to be fully available if

  $$avail(o) = True, \forall o \in O_s \wedge avail(p) = True, \forall p \in P_s$$

  A fully available service has all its entities available for the clients.

  Hence, the set of entities that contributes to the availability of the partial available system are the ones belonging to the subsets in the *minAvailable* that satisfies the availability assertion and is represented as:

  $$Entities_f = \bigcup_{\forall e \in P_s \cup O_s : avail(e) = True} e$$

- **Partial Availability** A service $s \in S$ in the system with entities $e \in P \cup O$ is said to be partially available if

  $$\exists M \in minAvailable(s), \forall e \in M : avail(e) = True$$

  A partially available service has some of its entities unavailable for the clients such that it still provides a part of the service or the full service at a lower rate.

  Hence, the set of entities that contributes to the availability of the partial available system are the ones belonging to the subsets in the *minAvailable* that satisfies the availability assertion and is represented as:

  $$Entities_p = \bigcup_{\exists M \in minAvailable(s), \forall e \in M : avail(e) = True} M$$

- **Unavailability** A service $s \in S$ in the system with entities $e \in P \cup O$ is termed to be unavailable if

  $$\forall M \in minAvailable(s), \exists e \in M : avail(e) = False$$

An unavailable service has all or some of its entities unavailable for the clients which result in no service being provided to the clients and is represented as S_u.

5.1 Calculating Value of Availability

A fully available service can be transformed into partially available service by deleting some entities required by the service, for example by killing a running process or by deleting objects with some value to the service.

Thus the value of *Availability* for a service in a system can be given in terms of fully available service and partially available service as

$$A = \frac{\sum_{e \in Entities_p} val(e)}{\sum_{e \in Entities_f} val(e)}$$

where $Entities_p$ and $Entities_f$ are the entities that contributes to the availability of partial and fully available systems respectively.

Chapter 6

MAPPING MODEL TO THE USE CASES

This section demonstrates the application of our model using various cases which are the part of the corpus created by Silowash, George, et al. (2012) for CERT, which archive incidents related to the insider attacks. The model representation for each case relates the subjects and actions of the case to the components of our model, and how the threat manifests in the form of the threat value.

The mitigation strategies for each case offer a set of possible countermeasures that could have been used to prevent the relevant attack. The suggested countermeasures provide a more technical approach as compared to the countermeasures provided in (Silowash, George, et al. 2012) which organizes the cases into different groups based on the best practices to better aid the organization with the design of an insider threat program. Although these provided solutions aides well for the policy-making, most of the practices remain irrelevant to the technical aspects of the system.

6.1 Case 1

6.1.1 Summary

Figure 6.1 represents a research chemist working on a research project of an organization involving electronic technologies. He used his computer system to copy data from the organization's server to a laptop located at competitive organization. In four months period, the insider downloaded 15,000 PDF files and more than 20,000 abstracts containing trade secrets of the victim organization. The leakage of data outside the system took place using a removable device as a physical medium of

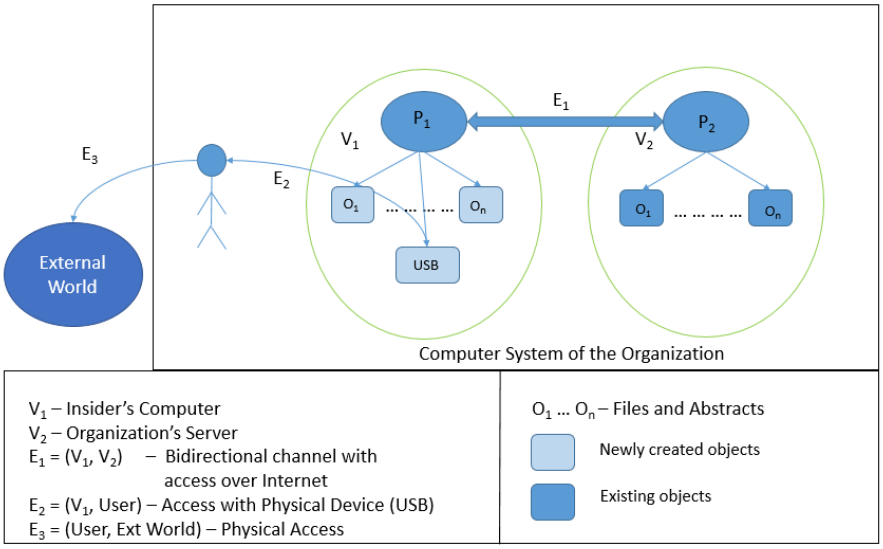

Figure 6.1: Model Representation of Case 1

transfer.

6.1.2 Model Representation

Using our model representation, we can view the transitions in the system. The sequence of transition steps are:

1. Insider created a process P_2 at V_2 using channel E_1 from his computer V_1, .
2. He accessed the objects at server V_2 through P_2.
3. With the use of 'read' and 'write' capability on objects, insider copied the objects from V_2 to V_1 through the bidirectional channel E_1.
4. He then used P_1 to transfer objects $o_1 ... o_n$ to a removable device through E_2.
5. Finally, he physically moved the removable device containing the objects outside the organization. This physical movement is represented by channel E_3.

6.1.3 Mitigation Strategy

1. **Threat Action -** *Large amount of data (around 300 documents) downloaded every day for a period of four months and copied to a laptop at a different organization.*

   Countermeasure - Network level access control should define a limit on the rate of data transfer for roles. A threshold should be defined to cap the maximum amount of data passing through a network point over a definite period of time.

2. **Threat Action -** *Accessed data not related to the research project.*

   Countermeasure - Consistent deployment of granular access control employing the principle of least privilege (insiders should have access only to the critical system functions and permissions necessary for carrying out their tasks) in the form of capabilities. Creation of regularly monitored audit log objects to capture access provided to high-risk users. Such an audit log would require a strategic balance between what kind of events can be logged without affecting the privacy of the insiders.

6.2 Case 2

6.2.1 Summary

Figure 6.2 represents a key member of IT support staff for a local bank who used his privileged physical access to the area running live system to insert a malicious xls file to bank's batch file transfer system when most of the bank's staff was off-duty. The malicious file executed an unauthorized transfer in favor of the insider. An xls spreadsheet was used to credit salary to the employees accounts through the bank's corporate account. The genuine salary transfer process was replicated in the malicious

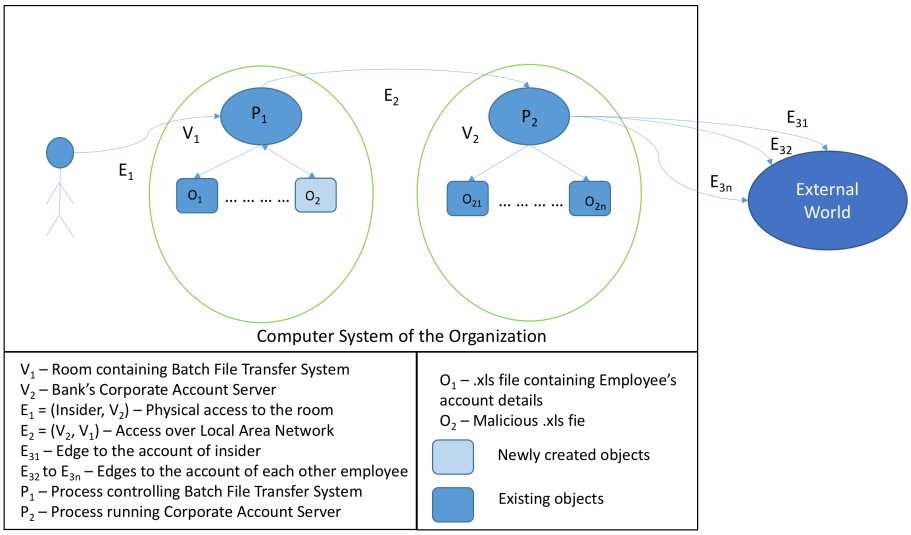

Figure 6.2: Model Representation of Case 2

file which had an additional list of charges, and this small amount was debited from the bank's corporate account and credited to his account each month.

6.2.2 Model Representation

The sequence of actions can be described with the model as:

1. Physical access for the insider to the area V_1 running the live systems to support the banking application system through high privileged capabilities.

2. He created procedure o_2 representing malicious .xls file to be executed along the normal .xls file which takes care of the salary transfer process.

3. Use of the edge E_2 by him to connect to the server V_2 containing the corporate accounts.

4. Creation of process P_2 at V_2 through P_1 by execution of procedure o_2 to transfer small amount from the bank's corporate account to insider's account.

6.2.3 Mitigation Strategy

1. **Threat Action -** *Accessed live system (meant to support banking application system) when most of the bank's staff were off-duty*

   Countermeasure - Access to vertices containing critical processes should require a combination of more than one capability (Separation of Duty) and the capabilities should enforce time-based access control. Separation of Duty is limited to organizations which can have more than one employee for similar roles.

2. **Threat Action -** *Created a new process inside a running critical system*

   Countermeasure - Creation of new process in a running system should not be allowed or some regularly monitored audit log objects should be created to capture the creation of a process that can access critical resources.

6.3 Case 3

6.3.1 Summary

Figure 6.3 represents a bookkeeper as an insider. She wrote around 70 checks, which were available for her access, to pay for herself in two years. She altered computer accounting records through her computer to show a different payee using the highly effective *write* capability to manipulate objects of nature that were critical to the organization's finance. $200,000 was embezzled from the organization.

6.3.2 Model Representation

The sequence of actions can be described with the model as:

1. Insider used physical access to the area V_1 containing physical checks and her computer.

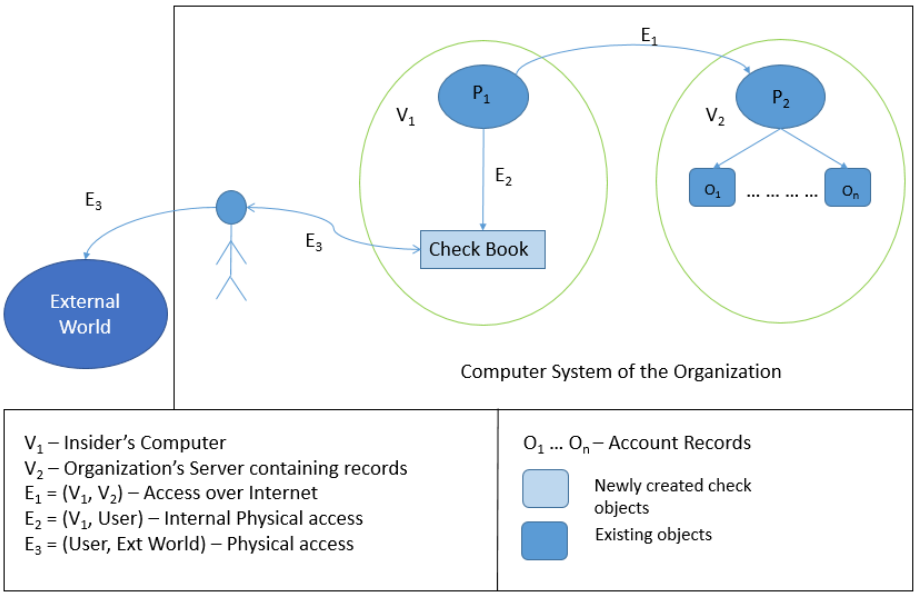

Figure 6.3: Model Representation of Case 3

2. She created the process P_1 to connect to the system containing the accounting records.

3. She used the edge E_1 to create process P_2 at V_2 and altering account objects o_1 to o_n using *write* capability.

4. Further, she used the edge E_2 to physically modify the checks.

5. She physically moved the check objects outside of the organization using edge E_3 to pay for herself.

6.3.3 Mitigation Strategy

1. **Threat Action -** *Wrote large amount of checks worth $200,000*

   Countermeasure -

   - Use of principle of Separation of Duty (splitting actions into separate duties and having multiple persons do each action in order to complete the task)

to perform actions such as finance operations that directly impact the organization. Two processes having the capabilities for the action should work together to perform it.

- A rate limit should be placed on the edge linking to the critical resource, e.g. the edge to checkbook in this case.

2. **Threat Action -** *Altered Organization's records to show a different payee*
   Countermeasure - Logical objects that represent a physical object of high value in the real world should have stronger access control. Creation of regularly monitored audit log objects to capture accesses through high-risk processes.

6.4 Case 4

6.4.1 Summary

Figure 6.4 represents the scenario where a Unix Engineer working as a contractor for a mortgage company was allowed to finish out the workday after being notified for the contract termination. He planted a logic bomb in a trusted script while being on-site. The script was designed to disable log-ins, delete root passwords and erase all data, including backup data, on those servers. The script was designed to remain dormant for 3 months but was detected before it could be executed.

6.4.2 Model Representation

The possible sequence of actions can be described with the model as:

1. Insider's access to his computer in the organization V_1 and creation of process P_1.

2. Further, he created a new process P_{21} to create a new procedure object (script) o_1 on V_2 which can be executed later.

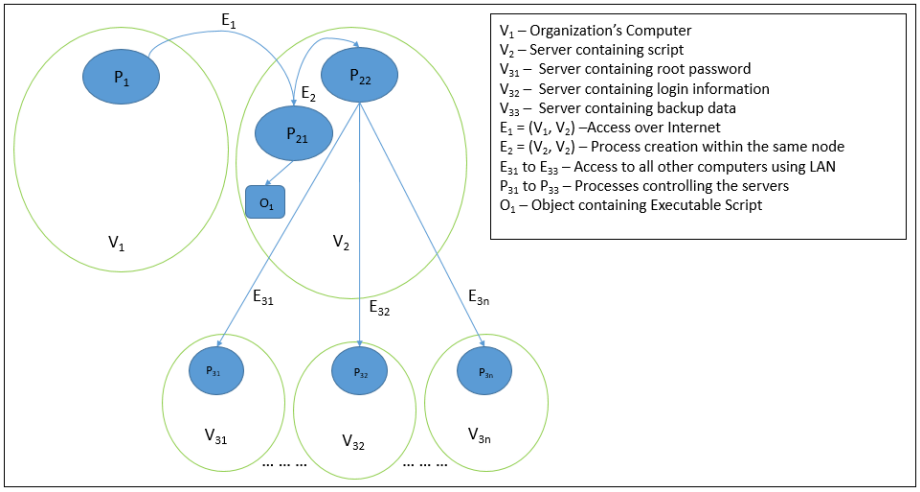

Figure 6.4: Model Representation of Case 4

3. Activation of the procedure o_1 would have created a process P_{22} which would further create processes on other nodes, using the *create* capability for the new processes.

4. P_{22} would have created edge E_{31} to create process P_{31} at V_{31}. P_{31} could delete all root passwords and hence disable log-in.

5. Similarly, P_{22} would have created edge E_{32} to create process P_{32} at V_{32}. P_{32} could delete back up data objects on the server.

6. Further P_3 would have created other process at different nodes to delete data, including backup data.

6.4.3 Mitigation Strategy

1. **Threat Action -** *Planted logic bomb in a trusted script*

   Countermeasure - Use of the principle of Separation of Duty for code development and review. The capabilities to edit the objects containing critical codes should be restricted to nodes meant for development and should be reviewed

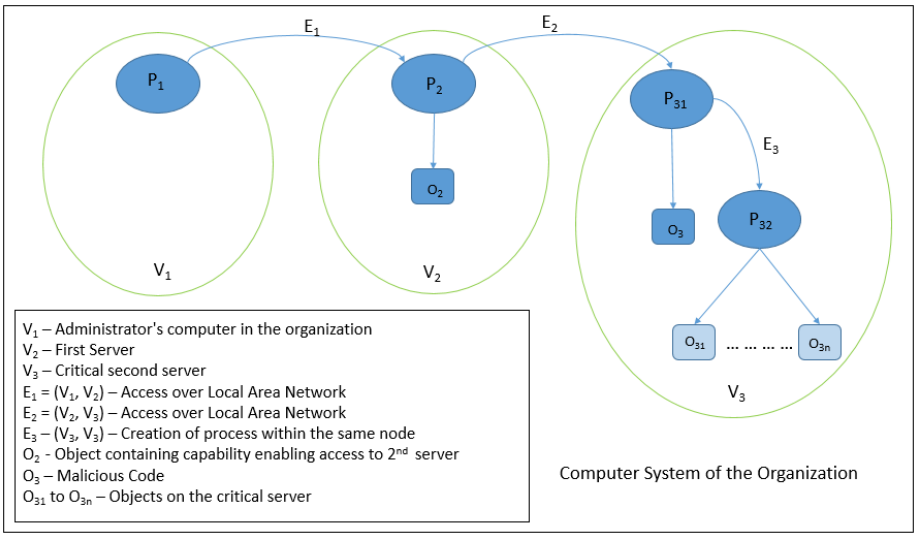

Figure 6.5: Model Representation of Case 5

before reaching the production environment.

2. **Threat Action -** *Script designed to disable log in and monitoring*

   Countermeasure - Putting critical processes such as logging and monitoring out of the application's purview. The edges between the nodes running scripts and the one controlling monitoring should have stronger access control.

6.5 Case 5

6.5.1 Summary

Figure 6.5 represents a scenario where a system administrator was responsible for critical system servers. Having root access to one server, he created a file that enabled him access to the second server. The insider then inserted a malicious code inside the second server that had capabilities to delete all organization's file when a certain volume is reached

6.5.2 Model Representation

The sequence of actions can be described with the model as:

1. Insider's access to his computer V_1 and creation of process P_1.
2. Insider then accessed first server at V_2 using E_1 over local area network which was created by root password as a capability.
3. He then created procedure object o_2 at V_2 with capability for creating process at V_3.
4. Further, he created process P_{31} at second server located at V_3 by creating an edge E_3 over local area network.
5. Once at V_3, he edited an object o_3 by inserting the malicious code using the '*write*' capability.
6. He planned to use procedure o_3 with malicious code owning capability to create a new process P_{32} at V_3 which has delete capabilities for objects at V_3
7. P_32 would have deleted objects o_{31} to o_{3n} using the *delete* capability.

6.5.3 Mitigation Strategy

1. **Threat Action -** *Use of privilege account to create an object enabling access to a different server*

   Countermeasure - Pulling entitlement decision (creation of capabilities) out of the application. Enabling Separation of Duty by moving policy-making decisions to a central node.

2. **Threat Action -** *Inserted malicious code that would delete all the organization's files.*

Countermeasure - Since the procedure would have found an access path (a sequence of one or more access points that lead to a critical system) to all the data in the organization, discovering all paths to the critical system can prevent such attacks. Access paths can be discovered by locating all the processes and the edges they can create based on the capabilities available. Regular monitoring for finding such access paths can reduce the number of unknown access paths available to the insiders.

3. **Threat Action -** *The code could have disabled system logging, removed history files and removed traces of malicious code*

   Countermeasure - Putting critical processes such as logging and monitoring out of the application's purview. Deletion capabilities should come under Separation of Duty access control with a threshold value on how much and how fast data can be deleted.

6.6 Case 6

6.6.1 Summary

Figure 6.6 has an insider working for the organization as an IT administrator. After being fired, he created another user account. Later he accessed the organization's server using the backdoor account from his home and public network. He deleted some customer data using this account and affected the availability of the company's server.

6.6.2 Model Representation

The sequence of actions can be described with the model as:

1. Creation of process P_1 by insider using access to the computer V_1.

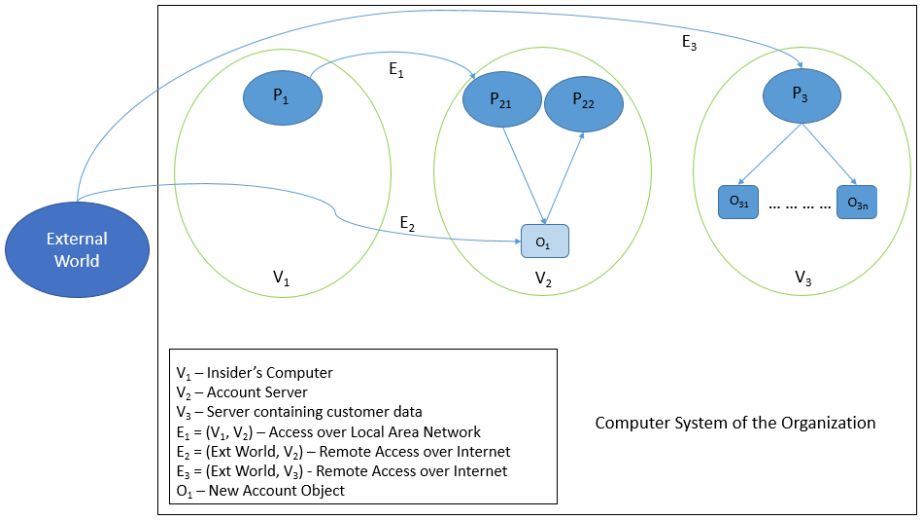

Figure 6.6: Model Representation of Case 6

2. He accessed the account server at V_2 using E_1 to create a process P_2.

3. He then created an account object at account server using P_2.

4. Further, he gained capabilities through the newly created account, after creating an edge E_2 using home computer to V_2.

5. Insider used newly gained capabilities to create a new edge E_3 from outside the system to V_3

6. Finally he deleted objects o_{31} to o_{3n} using the delete capabilities.

6.6.3 Mitigation Strategy

1. **Threat Action -** *Creation of new backdoor account with administrator privilege*
   Countermeasure - The nodes representing Policy Enforcement Points (Point which manages access authorization policies) should be kept out of the business application nodes, with additional enabled separation of duty. Such a separation could be expensive and organizations with limited resources can just do with

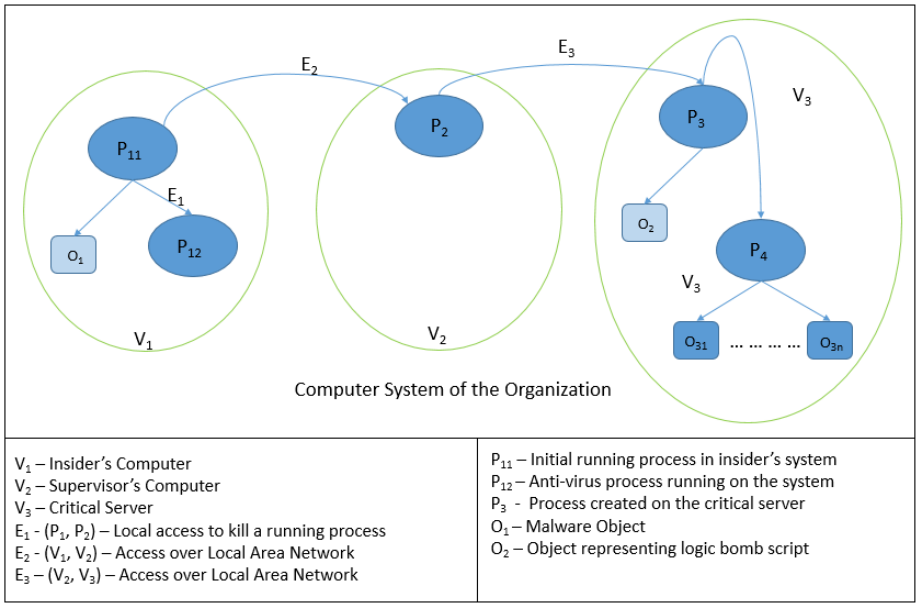

Figure 6.7: Model Representation of Case 7

the separation of duty.

2. **Threat Action -** *Deleted customer data and made it hard for customers to access data*

   Countermeasure - New account created by the person had capability to delete data which is a case of capability creation. Capability creation should be entitled to Separation of Duty principle.

6.7 Case 7

6.7.1 Summary

Figure 6.7 describes a scenario where insider disabled the anti-virus application in the organization's system. Following it, a malware was installed to it which was used to gain unauthorized access to his supervisor's system. This access was then used to plant a logic bomb at a critical server.

6.7.2 Model Representation

The sequence of actions can be described with the model as:

1. Insider used the access to his computer V_1 for creation of process P_{11}.
2. He disabled the anti virus process P_{12} by killing it using the *kill* capability through the edge E_1.
3. He created a malware object o_1 at V_1 as no anti-virus was available to stop this event.
4. Further, he used an edge E_2 through internet from V_1 to supervisor's computer V_2 using *access* capability gained from the newly created malware.
5. Creation of process P_2 by him at supervisor's computer which was further used to create an edge E_3 from V_2 to a critical server V_3.
6. Next he created a procedure object o_2 at V_3 which could act as a logic bomb.
7. o_2 could have been executed to create the process P_4 at server V_3 which would have affected other objects in the server using *write* or *delete* capability.

6.7.3 Mitigation Strategy

1. **Threat Action -** *Disabled anti-virus application to install a malware*
   Countermeasure - Critical process such as anti-virus should be centrally controlled rather than having the control dispatched to local levels. Systems must also have certain "choke-points" (nodes through which data must flow during system operations) which can provide filtering on malicious codes and software.

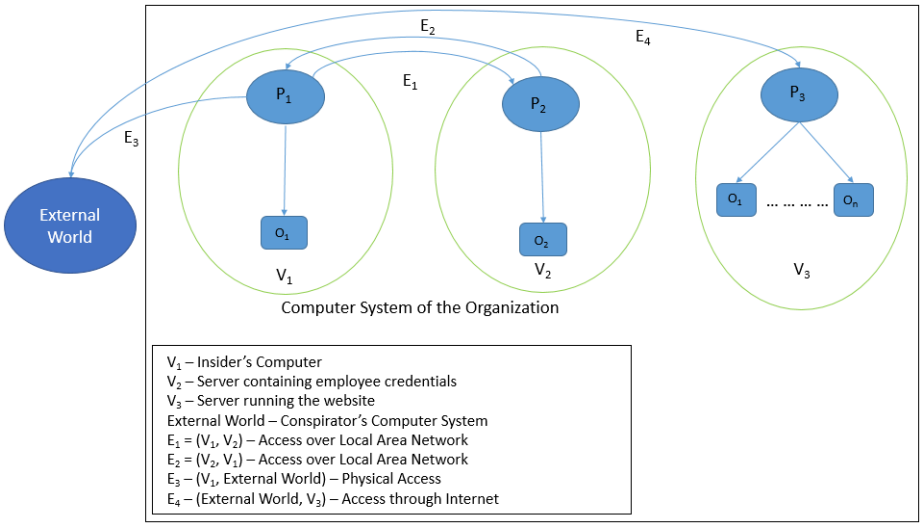

Figure 6.8: Model Representation of Case 8

2. **Threat Action -** *Used malware to gain access to the supervisor system*

   Countermeasure - Creation of capability should not be allowed by any process. Such access rights should be limited to certain highly effective process.

6.8 Case 8

6.8.1 Summary

Figure 6.8 describes a computer system belonging to a large telecommunication firm where the insider worked as a help desk technician. The insider installed hacking tools in his computer, stole other employees credential from the server and passed it to an external conspirator who used this to gain unauthorized access to the website of the organization and deface it. This led to a loss in customer and share value for the organization.

6.8.2 Model Representation

The sequence of actions can be described with the model as:

1. The insider accessed the computer V_1 and created the process P_1 to install the hacking tool as an procedure object o_1.

2. He used o_1 to create a new process P_2 at the server V_2 which contained other employee credential using Local Area Network represented by edge E_2.

3. He then passed the credential to the external agent physically through edge E_3.

4. External agent created an edge E_4 to the vertex V_3 which contained the website of the organization.

5. He used the write capabilities gained through the credentials to edit the objects (o_1 to o_n) stored in the website server.

6.8.3 Mitigation Strategy

1. **Threat Action -** *Help Desk technician installed hacking tools at the company's computer*

   Countermeasure - Trusted system should be highly resistant to manipulation from within by prohibiting the creation of a new process in the system without required capabilities.

2. **Threat Action -** *Passed credential to external conspirator*

   Countermeasure - The access paths to critical nodes such as those containing credentials should be protected with stronger capabilities. Capabilities should not be transferable outside the organization. This is not applicable for the organizations whose systems are controlled by people outside the organizations, such as contract agencies, in which case such a transfer is inevitable and stronger access control on external capabilities could facilitate with the security.

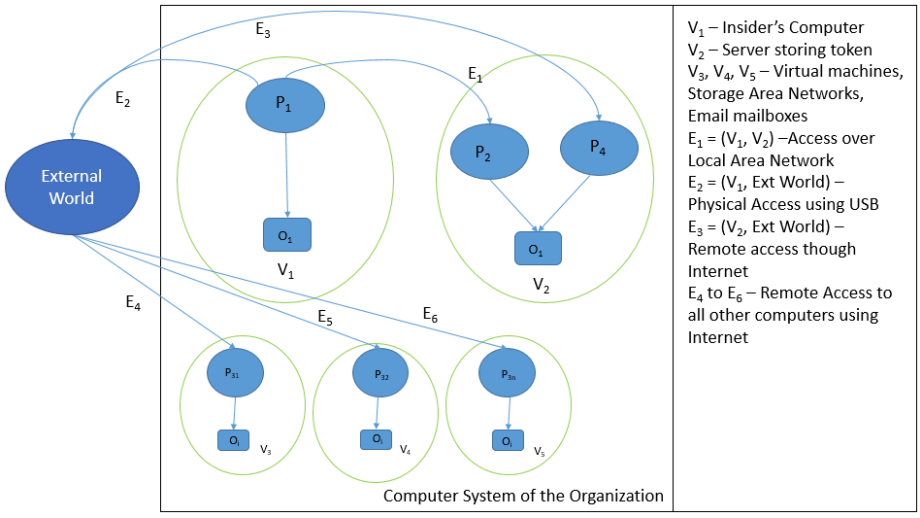

Figure 6.9: Model Representation of Case 9

6.9 Case 9

6.9.1 Summary

Figure 6.9 describes a network engineer working for a retail organization who used USB VPN tokens for remote access. On being terminated, the insider created a token in the name of a fake employee, contacted insider IT department to convince them to activate the token. He further used it to access the network and delete virtual machines, email servers and storage area network.

6.9.2 Model Representation

The sequence of actions can be described with the model as:

1. The insider created process P_1 at his own computer V_1 inside the organization.
2. He used P_1 to create a VPN token object o_1 and passing it to the V_2, the server containing all the tokens.
3. Token was carried outside the organization physically by him through USB,

represented by edge E_2.

4. He then contacted the IT department through edge E_3 representing internet connection, which activated the token object o_1 by using the process P_4.

5. Further, he used the activated token and its capability to access various servers to create a new process at servers such as email and SAN server.

6. Finally he used the *delete* capability to delete the objects in the newly created processes on the servers.

6.9.3 Mitigation Strategy

1. **Threat Action -** *Created an USB VPN token in name of a fake employee*
   Countermeasure - Hardware authentication objects should only be forgeable by the process with strong capabilities.

2. **Threat Action -** *Convinced IT department to activate the token*
   Countermeasure - Request edges from outside the organization should be separable from the one within the organization. Since activation of tokens led to the creation of capabilities, it should involve the Separation of Duty principle.

3. **Threat Action -** *Several months later, VPN used to delete virtual machines, shutdown SAN and delete email boxes.*
   Countermeasure - Implementation of procedure to be used for removal of employees with a high degree of access (e.g. that doesn't allow them to retain such privileges after a certain point of time) by implementing the principle of session-based access control. The killing of a critical process such as the one running SAN should be bound by strong authentication rules.

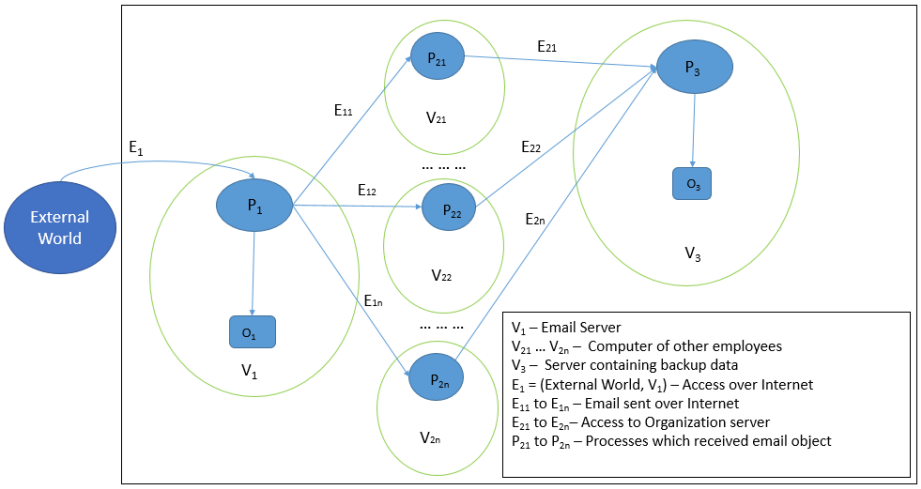

Figure 6.10: Model Representation of Case 10

6.10 Case 10

6.10.1 Summary

Figure 6.10 describes a system administrator working for a unified messaging service discovered a security vulnerability in the organization's email service. Six months after leaving the organization, the insider used the organization's email account to email 5600 of organization's customer. The email exposed the security flaw and directed them to a link which crashed the victim organization's servers and caused a major damage to reputation.

6.10.2 Model Representation

The sequence of actions can be described with the model as:

1. Creating process P_0 at his own computer outside the organization.

2. Using P_0 to create a process P_1 on the email server inside the organization.

3. Using the email server and write capability to create email objects $o_1 ... o_n$ and

send it to various other users.

4. Other users accessing the email object and following the link to access the resource on the server at V_3.

5. Overloading the process P_3 at V_3 with multiple requests, thus crashing the server.

6.10.3 Mitigation Strategy

1. **Threat Action -** *After leaving the organization, the old account used to email employees.*

   Countermeasure - Account invalidation should cause removal of all the capabilities associated with the account.

2. **Threat Action -** *Emailed 5600 customers using the organization's email account.*

   Countermeasure - The rate of creation of objects (email in this case) should be limited and bound by a threshold and a dynamic warning should be generated when the threshold is exceeded.

6.11 Case 11

6.11.1 Summary

Figure 6.11 describes a situation where a tax preparer for a tax preparation service. During working hours, the insider printed Personal Identification Information (PII) of 30 customers. This information was used by the insider to submit fraudulent tax returns. The refunds of around $290,000 were deposited by the insider at various bank accounts.

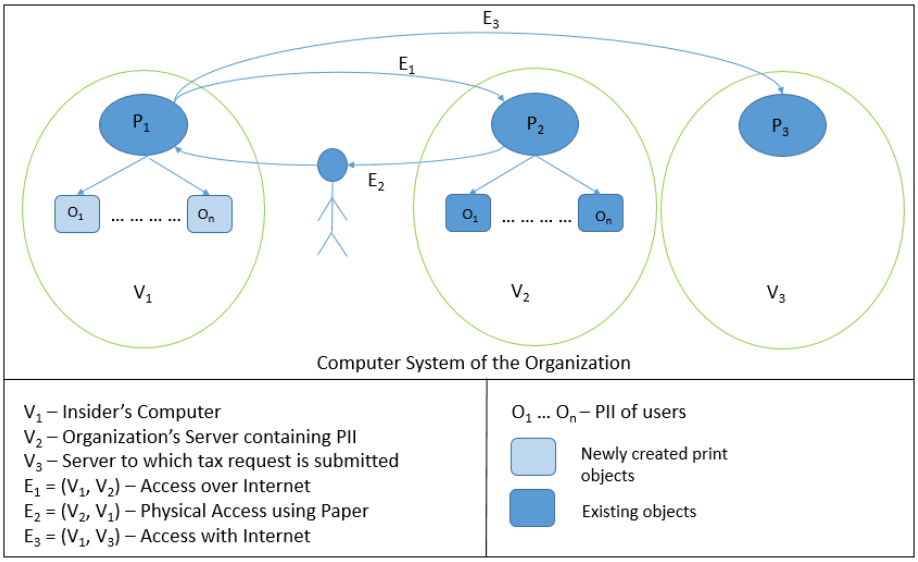

Figure 6.11: Model Representation of Case 11

6.11.2 Model Representation

The sequence of actions can be described with the model as:

1. Creating process P_1 at his own computer V_1 inside the organization.
2. Using P_1 to create a process P_2 on the server containing the PII of the customers.
3. Printing the PII objects $o_1 ... o_n$ with the 'Read' capability.
4. Physically accessing the printed objects and moving them to his own space represented by edge E_2.
5. Creating the process P_3 to submit fraudulent tax returns using the SSN in the printed PII objects.
6. Receiving the funds in his own bank accounts.

6.11.3 Mitigation Strategy

1. **Threat Action -** *Insider printed PII of 30 customers during work hours.*
   Countermeasure -Actions such as printing should be defined as a 'Copy' capability and should have more granular access control for critical objects.

2. **Threat Action -** *Submit fraudulent tax records with stolen SSN*
   Countermeasure - Management of critical objects should be handled with the principle of Separation of Duty.

6.12 Case 12

6.12.1 Summary

Figure 6.12 describes a scenario where insider planned for the organization and its customer to loose $1 Million over the course of one year by tempering the risk assessment's program's code. The risks of deal increments were kept very small to prevent any suspicion so that the traders would not be able to realize the deals getting riskier with the time.

6.12.2 Model Representation

The sequence of actions can be described with the model as:

1. Creating process P_1 at his own computer while working inside the organization.

2. Creating process P_2 at the server containing the risk assessment program.

3. Manipulating the risk management program O_{21} using 'write' capability.

4. Using an edge E_2 at V_2, to create the process P_3 to run the edited procedure object.

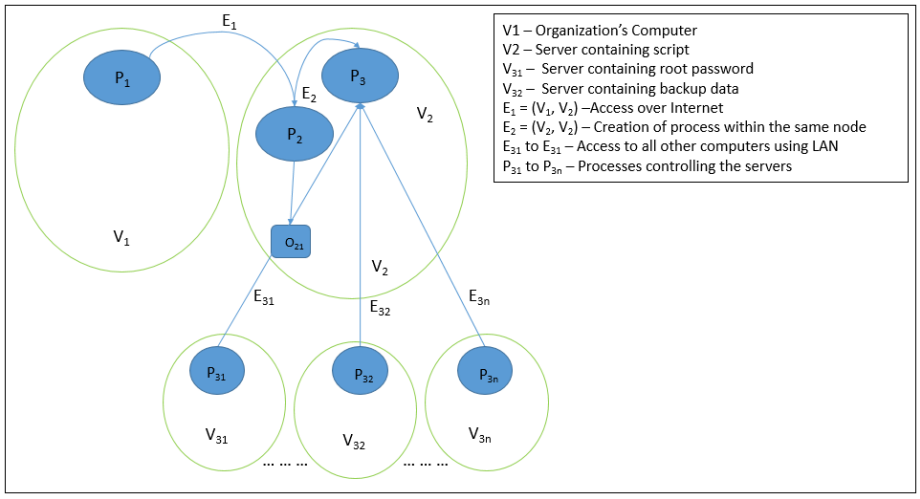

Figure 6.12: Model Representation of Case 12

5. Accessing of process P_3 running the riskier deal program by various trader's processes ($P_{31}...P_{3n}$) at their respective vertices using the execute capability.

6.12.3 Mitigation Strategy

1. **Threat Action -** *Insertion of logic bomb in the risk assessment program.*

   Countermeasure - Configured audit log to capture the changes made in critical procedures which are accessed by high-risk users. This separate log objects should be monitored frequently.

6.13 Case 13

6.13.1 Summary

Figure 6.13 describes an insider working as a lead software engineer for a government agency, who on learning about his future demotion and reduction in pay, wrote code in an obscure way to undermine the project's transition. The insider copied the source code to a removable media and encrypted it with a password. He also deleted the source code from his laptop and it later turned out to be the only copy of the

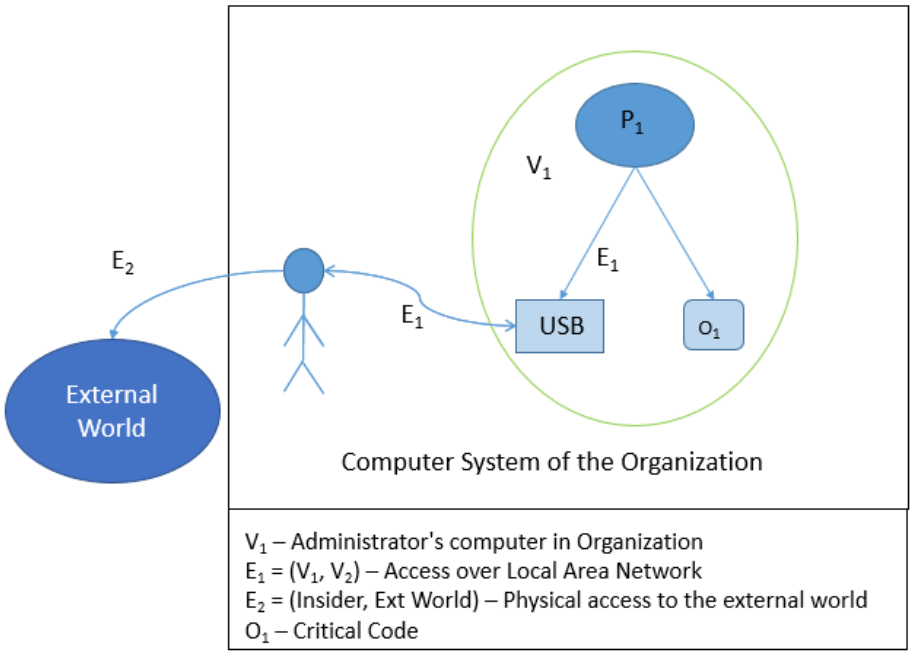

Figure 6.13: Model Representation of Case 13

source code for the system.

6.13.2 Model Representation

The sequence of actions can be described with the model as:

1. The insider accessed the code in procedure object o_1 on his computer V_1 inside the organization using the process P_1.

2. He used the $'write'$ capability to write obscure code in the procedure, rendering it unreadable.

3. Moreover, he used removable media to copy the code objects from the computer to the device.

4. He then used the edge E_1 to move the physical media from V_1 to the external world.

5. Lastly he deleted the only copy of source code object from the node V_1.

6.13.3 Mitigation Strategy

1. **Threat Action -** *Wrote critical source code in an obscure way*

   Countermeasure - *write* capability for critical objects should be reviewed before making its way towards production. Even for a very small organization with limited work-force, such capabilities should involve more than one process.

2. **Threat Action -** *Use of removable media to transfer code to the external world*

   Countermeasure - External devices can provide channels with the unlimited rate for data extrusion. Such channels, if allowed, should have a capacity limit and all such transfer should require extra capability.

3. **Threat Action -** *Deleted the only copy of the source code*

   Countermeasure - Objects of type procedures should have more than just local copy and should be submitted to a server to maintain software baselines.

6.14 Case 14

6.14.1 Summary

Figure 6.14 describes a software developer and tester as an insider who was terminated for poor performance but the victim organization failed to change a shared account password upon his departure. The company's laptop provided by his subsequent employer was used by him to remotely access 14 of the victim organization's user accounts. The insider also exploited 13 systems storing trade secrets valued at approximately $1.3 million.

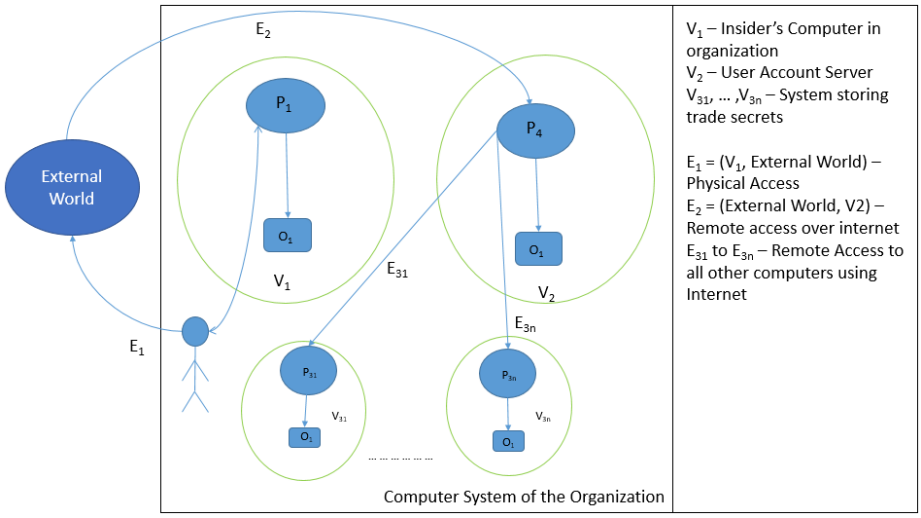

Figure 6.14: Model Representation of Case 14

6.14.2 Model Representation

The sequence of actions can be described with the model as:

1. The insider accessed his own computer at V_1 to learn the shared password which provided him the capability to access the computer system remotely.

2. He physically moved this capability to the external organization's system using the edge E_1.

3. The credential of another employee along with the shared password capability was used by him to access user account server at V_2.

4. Further, he accessed accessing 13 other systems containing trade secrets ($V_{31}...V_{3n}$).

6.14.3 Mitigation Strategy

1. **Threat Action -** *Use of shared password for shared accounts.*

   Countermeasure - Separate capabilities should be used for access to the

shared accounts. Such separation can help ease of removal of capabilities when a process is removed.

2. **Threat Action -** *An access from the external world was able to exploit 13 systems.*

   Countermeasure -

   - A limit should be established on a number of processes that could be created from a single access.
   - Channels from the external world should have restricted capabilities and access through such channels should always be monitored.

6.15 Limitations

The ability to predict the above attacks and their damage value based on the model and the algorithms have inherent limitations. For some of the above cases, it is difficult to foresee the attacks through the model as attackers keep coming up with unique attack approaches and such approaches could be beyond the scope of policy framers. Consider the examples in Case 2 and Case 12. Here, the attackers in both the cases tried to modify procedures to change the behavior of the process which led to revenue losses to the organizations. Although the model is capable of representing such attack after it has taken place, it is difficult to anticipate such attacks and calculate the damage in advance. Similarly, Case 3 represents a scenario of physical manipulation of objects and it is tough to calculate the extent of such manipulation beforehand. Hence, our algorithm for data leakage cannot take into account such leakage as the capacity of physical channels can be unpredictable.

In cases where social factors are used to devise attacks, there are limited means to model such events. Such as in Case 9, where convincing the IT department to activate

the token can be seen as an attack involving social engineering. Likewise, the extent of external help is also difficult to be determined, and hence could not be modeled perfectly before the attack occurs. Case 10 is an example of such a scenario where the insider used the vulnerability of the organization's customers to attack the services. While cases like these do arise, but from our study of several attacks, we found that such method of attacks are rare. But such parameters should be considered while developing the policies and appropriate methods should be engineered to prevent them.

Chapter 7

CONCLUSION

As organizations continuously face threats from users of their computer system, different models have been introduced to represent these threats and predict the risk value of a user. In this thesis, we take these approaches further by looking at the problem from a perspective of capabilities that the users possess, and how the calculation of the potential threat of that user depends on dynamic augmentation of these capabilities. Although we focus on two major aspects of damage, i.e. information flow and leakage, and change in the availability of the system, yet we believe that the model serves a more general purpose for almost every kind of damage that a user can cause to the computer system.

We implement the model to show that it can be used in real-world scenarios. We also analyze several insider attacks that took place in the last few decades to suggest how our model could represent those attacks and then suggest mitigation strategies within the scope of our model for all such use cases.

There are several areas in which the model can be improved in the future.

- The current model represents the physical users as a process with pre-defined capabilities. But the physical capabilities of a user can be dependent on many other features of the external environment and would need more detailed representation.

- The thesis focuses on the damage potential of a user of the system based on just their individual capabilities. However, the user can collude with other users to complete its attack. Such collusion can dynamically change the damage

potential of the insider and can be a subject of future research.

- The suggested mitigation approaches for all the use cases are within the scope of our model. Other mitigation approaches can be suggested keeping other real-world factors in mind.

REFERENCES

Ammann, Paul, Duminda Wijesekera, and Saket Kaushik. "Scalable, Graph-based Network Vulnerability Analysis." Proceedings of the 9th ACM Conference on Computer and Communications Security - CCS 02, 2002. doi:10.1145/586139.586140.

Anderson, Robert H., and Richard C. Brackney. "Understanding the Insider Threat: Proceedings of a March 2004 Workshop." Santa Monica, CA: RAND, 2004.

Artz, Michael Lyle. NetSPA: A Network Security Planning Architecture. Master's thesis, 2002.

Baracaldo, Nathalie, and James Joshi. "An Adaptive Risk Management and Access Control Framework to Mitigate Insider Threats." Computers and Security 39 (2013): 237-54. doi:10.1016/j.cose.2013.08.001.

Bishop, Gates. "Defining the Insider Threat." UC Davis Previously Published Works, UC Davis, 2008

Celikel E, Kantarcioglu M, Li X, Bertino E. "A risk management approach to RBAC." Risk and Decision Analysis 2009;1(2):21e33.

Chen L, Crampton J. "Risk-aware role-based access control." In: Proc. of the 7th International Workshop on security and trust management 2001.

Chinchani, R., A. Iyer, H.q. Ngo, and S. Upadhyaya. "Towards a Theory of Insider Threat Assessment." 2005 International Conference on Dependable Systems and Networks (DSN05), 2005. doi:10.1109/dsn.2005.94.c

Cohen, Ellis, and David Jefferson. "Protection in the Hydra Operating System." ACM SIGOPS Operating Systems Review 9, no. 5 (1975): 141-60. doi:10.1145/1067629.806532.

Colwill, Carl. "Human Factors in Information Security: The Insider Threat ? Who Can You Trust These Days?" Information Security Technical Report 14, no. 4 (2009): 186-96. doi:10.1016/j.istr.2010.04.004.

Gorodetski, Vladimir, and Igor Kotenko. "Attacks against Computer Network: Formal Grammar-Based Framework and Simulation Tool." Lecture Notes in Computer Science Recent Advances in Intrusion Detection, 2002, 219-38. doi:10.1007/3-540-36084-0_12.

Greitzer, Frank L., and Deborah A. Frincke. "Combining Traditional Cyber Security Audit Data with Psychosocial Data: Towards Predictive Modeling for Insider Threat Mitigation." Insider Threats in Cyber Security Advances in Information Security, 2010, 85-113. doi:10.1007/978-1-4419-7133-3_5.

Ha, D., S. Upadhyaya, H. Ngo, S. Pramanik, R. Chinchani, and S. Mathew. "Insider Threat Analysis Using Information-Centric Modeling." Advances in Digital Forensics III IFIP ? The International Federation for Information Processing, 2007, 55-73. doi:10.1007/978-0-387-73742-3_4.

Harel, Amir, Asaf Shabtai, Lior Rokach, and Yuval Elovici. “M-Score: A Misuseability Weight Measure.” IEEE Transactions on Dependable and Secure Computing 9, no. 3 (2012): 414-28. doi:10.1109/tdsc.2012.17.

Jajodia, Sushil, Steven Noel, and Brian O?Berry. “Topological Analysis of Network Attack Vulnerability.” Managing Cyber Threats Massive Computing, 2003, 247-66. doi:10.1007/0-387-24230-9_9.

Lippmann, R. P., and K. W. Ingols. “An Annotated Review of Past Papers on Attack Graphs.” Lincoln Laboratory, Massachusetts Institute of Technology, 2005. doi:10.21236/ada431826.

Phillips, C., T. Gaylor, and L.p. Swiler. “A Graph-based Network-vulnerability Analysis System.” 1998. doi:10.2172/573291.

Porras, P. A., P.G. Neumann. “EMERALD: Event Monitoring Enabling Response to Anomalous Live Disturbance”. In Proceedings of the 19th National Computer Security Conference, Pages 22-25, Baltimore, Maryland, Oct 1997. National Institute of Standards and Technology (NIST) / National Computer Security Center (NCSC).

PwC. “Key Findings from the 2015 US State of Cybercrime Survey.” PwC. Accessed November 23, 2018. https://www.pwc.com/us/en/services/consulting/library/us-cybercrime-survey-2015.html.

PwC. “2017 U.S. State of Cybercrime Highlights.” SEI Insights. January 17, 2018. Accessed November 24, 2018. https://insights.sei.cmu.edu/insider-threat/2018/01/2017-us-state-of-cybercrime-highlights.html.

Salim F, Reid J, Dawson E, Dulleck U. “An approach to access control under uncertainty.” In: Availability, reliability and security (ARES), 2011 Sixth International conference on 2011. p. 1e8. http://dx.doi.org/10.1109/ARES.2011.11.

Schultz, Eugene. “A Framework for Understanding and Predicting Insider Attacks.” Computers and Security, vol. 21, no. 6, 2002, pp. 526?531., doi:10.1016/s0167-4048(02)01009-x.

Sheyner, O., J. Haines, S. Jha, R. Lippmann, and J.m. Wing. “Automated Generation and Analysis of Attack Graphs.”Proceedings 2002 IEEE Symposium on Security and Privacy. doi:10.1109/secpri.2002.1004377.

Silowash, George, Dawn Cappelli, Andrew Moore, Randall Trzeciak, Timothy J. Shimeall, and Lori Flynn. “Common Sense Guide to Mitigating Insider Threats 4th Edition.” 2012. doi:10.21236/ada585500.

Stanifor-Chen, S., S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, D. Zerkel. “GRIDS - A Graph Based Intrusion Detection System for Large Networks”. In Proceedings of the 20th National Information System Security Conference (NISSC), Oct. 1996. National Institute of Standards and Technology (NIST), 1996.

Swiler, L.p., C. Phillips, D. Ellis, and S. Chakerian. "Computer-attack Graph Generation Tool." Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX01, June 2001. doi:10.1109/discex.2001.932182.

Theoharidou, Marianthi, Spyros Kokolakis, Maria Karyda, and Evangelos Kiountouzis. "The Insider Threat to Information Systems and the Effectiveness of ISO17799." Computers and Security 24, no. 6 (2005): 472-84. doi:10.1016/j.cose.2005.05.002.

Wood B, "An insider threat model for adversary simulation", SRI Int. Res. Mitigating Insider Threat Inf. Syst., vol. 2, 2000, pp. 1-3.

Zhang, G., J. Li, G. Gu. "Research on Defending DDoS Attack-an Expert System Approach". In Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics, Netherlands, Vol. 4, pp. 3554-3558, Oct. 2004.