A Layered Graphical Model for Cloud Forensic Mission Attack Impact Analysis

Changwei Liu, Anoop Singhal, Duminda Wijesekera

Chapter 15

A LAYERED GRAPHICAL MODEL FOR CLOUD FORENSIC MISSION ATTACK IMPACT ANALYSIS

Changwei Liu, Anoop Singhal and Duminda Wijesekera

Abstract Cyber attacks on the systems that support an enterprise's mission can significantly impact its objectives. This chapter describes a layered graphical model designed to support forensic investigations by quantifying the mission impacts of cyber attacks. The model has three layers: (i) an upper layer that models operational tasks and their interdependencies that fulfill mission objectives; (ii) a middle layer that reconstructs attack scenarios based on the interrelationships of the available evidence; and (iii) a lower level that uses system calls executed in upper layer tasks in order to reconstruct missing attack steps when evidence is missing. The graphs constructed from the three layers are employed to compute the impacts of attacks on enterprise missions. The National Vulnerability Database – Common Vulnerability Scoring System scores and forensic investigator estimates are used to compute the mission impacts. A case study is presented to demonstrate the utility of the graphical model.

Keywords: Mission attack impact, cloud forensic analysis, layered graphical model

1. Introduction

Organizational missions that abstract activities envisioned by organizations are usually defined at the high-level as a collection of business processes. Cyber attacks on enterprise infrastructures that support such missions can cause significant impacts. Meanwhile, a growing number of business processes and services are being hosted by cloud operator data centers. Given that most network infrastructures, including cloud infrastructures, rely on hardware and software assets, attacks that target these assets could significantly impact the missions they support.

Therefore, analyzing and quantifying the mission impacts of cyber attacks are very important to infrastructure risk managers who seek to mitigate security threats and improve mission resilience.

NIST's National Vulnerability Database – Common Vulnerability Scoring System (NVD-CVSS) provides impact estimates of exploitable vulnerabilities in information technology systems [8]. Several approaches use the Common Vulnerability Scoring System to predict the impacts of multi-step attacks on assets by considering all possible attack paths [7, 9, 15]. However, evaluating all the attack paths is infeasible for a forensic investigator intending to assess the damage because of the large number of attack paths generated when all possible vulnerabilities are considered. Additionally, the scoring system only considers publicly-reported vulnerabilities, not zero-days.

Because post-attack artifacts obtained during forensic investigations provide information that can be used to analyze attacks, the proposed layered graphical model uses this information to quantify the impacts of attacks on organizational missions. The graphical model comprises three layers and two mapping algorithms. The upper layer models operational tasks and their interdependencies that constitute the final mission as a collection of choreographed tasks. The middle layer collects evidence from intrusion detection systems and event logs to reconstruct attack scenarios. The lower layer reconstructs potentially missing attack steps using system calls executed to fulfill the upper layer tasks; this is required when evidence needed to reconstruct the attack scenarios in the middle layer is not available. Finally, the two mapping algorithms integrate the information obtained from the three layers to ascertain how mission execution was impacted during the attacks.

The three layers of graph-like dependency information provide a means to compute attack impacts on organizational missions using the National Vulnerability Database – Common Vulnerability Scoring System or forensic investigator estimates. A case study is presented to demonstrate the utility of the graphical model, in particular, how it can be used to migrate attack risks in network infrastructures, including those supporting cloud services. The model is unique because it provides an integrated forensic analysis framework that quantifies the mission impacts of multi-step attacks in complex enterprise infrastructures.

2. Background and Related Work

This section briefly discusses cloud forensics and related research.

2.1 Cloud Forensics

Digital forensic investigators seek evidence of attack activities on computers and networks. Evidence on a computer typically resides in physical memory or on the hard disk; the evidence may be recovered using imaging and data analysis tools [13]. Evidence from a network is typically obtained in the form of network traffic capture files. Some network forensic tools, such as Snort, are also used for intrusion detection.

In the case of cloud environments, NIST [6] has defined deployment models such as software-as-a-service (SaaS), platform-as-a-service (PaaS) and infrastructure-as-a-service. Software-as-a-service enables clients to use service provider applications running on a cloud infrastructure. Platform-as-a-service enables clients to deploy cloud client applications that use programming languages, libraries, services and tools supported by a cloud provider. Infrastructure-as-a-service provides clients with the ability to provision processing, storage, networks and other computing resources.

According to Ruan et al. [12], cloud forensics is a subset of network forensics because it follows the main phases of network forensics, albeit with techniques tailored to cloud computing environments. Evidence acquisition is different in a software-as-a-service deployment compared with an infrastructure-as-a-service deployment. In the case of a software-as-a-service deployment, a forensic investigator depends entirely on the cloud service provider. In contrast, in an infrastructure-as-a-service deployment, an investigator can acquire evidence from virtual machine images that execute on client computer systems.

2.2 Related Work

Attackers tend to use multi-step, multi-stage attacks to bypass security countermeasures and impact important business services. Several researchers have proposed models for estimating the mission impacts of such attacks by considering all known vulnerabilities. Sun et al. [16] have proposed a multi-layer impact evaluation model for estimating mission impacts. At the bottom is a vulnerability layer that maps to an asset layer, then to a service layer and finally to the top mission layer. This layered design enables mission impacts to be computed using the National Vulnerability Database – Common Vulnerability Scoring System scores and the relationships between missions and lower-level vulnerabilities. However, the model does not incorporate a methodology for constructing attack paths.

Other researchers [15] have combined mission dependency graphs with attack paths generated by the MulVAL attack graph generation tool [11]

to estimate the mission impacts of attacks on cloud infrastructures. Noel et al. [9] have designed a cyber mission impact assessment framework that analyzes all attack paths leading to attack goals to evaluate potential mission impacts; their framework leverages the Business Process Modeling Notation (BPMN) and a topological-vulnerability-analysis-based attack graph generation tool [2]. However, these approaches depend on attack paths constructed from vulnerability information provided by the bug report community (including NIST's National Vulnerability Database – Common Vulnerability Scoring System) to assess the impacts of attacks. As a result, the approaches do not scale to large infrastructures and cannot handle zero-day attacks.

Digital forensic researchers have employed post-attack evidence and correlation rules to reconstruct attack scenarios in investigations of criminal activities and enterprise incidents [4, 17]. For example, Liu et al. [3] have integrated the MulVAL Prolog logic-based tool with a vulnerability database and an anti-forensics database to ascertain the admissibility of evidence and explain missing evidence caused by anti-forensic activities [3]. Liu et al. [5] have also extended this work by using system calls to reconstruct attack scenarios in which certain attack steps cannot be determined due to missing evidence. However, no research in digital forensics has focused on assessing the mission impacts of attacks on enterprise infrastructures.

3. Graphical Model

Figure 1 shows the layered graphical model for mission impact evaluation. The lower layers reconstruct attack paths, enabling attacks to be mapped to tasks and missions in the upper layer in order to compute the mission impact.

3.1 Upper Layer

The upper layer of the model represents tasks and missions in terms of business processes and connects them to business process diagrams (BPDs) using the Business Process Modeling Notation. Components such as tasks, events, sequencing, exclusive choices, parallel gateways, message flows and pools [1] are used to construct business process diagrams.

Definition 1. (Business Process Diagram): A business process diagram is a five-tuple $(Pool, T, E, C, OP)$ that satisfies the following conditions:

- T is a set of tasks and E is a set of events.

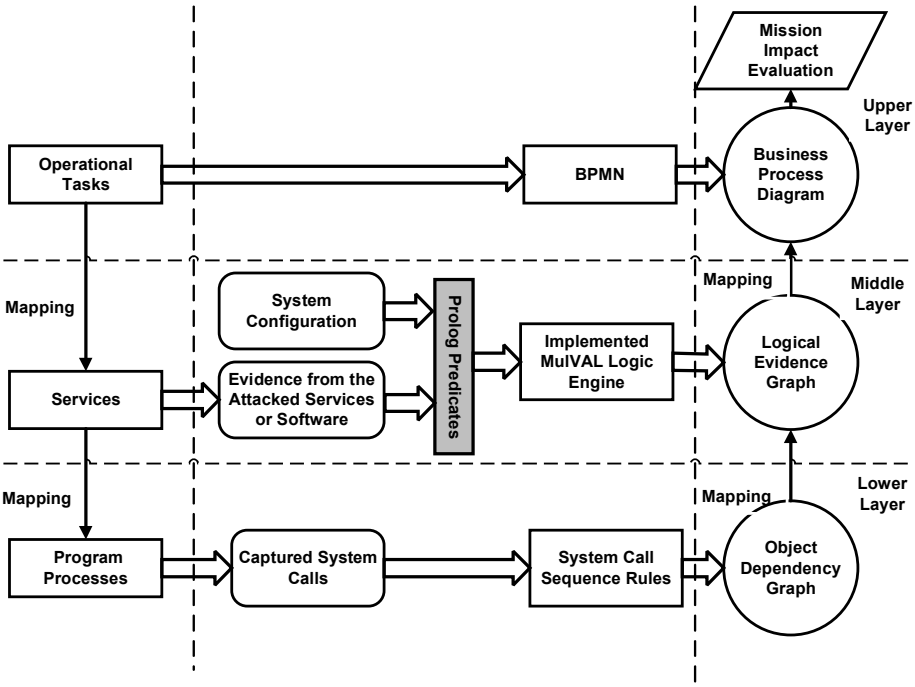


Figure 1. Layered graphical model for mission impact evaluation.

- Let E^{send}, E^{rec} be disjoint subsets of E and let $e_{start}, e_{end} \in E$ be the start and end events, respectively. Then, the set of events $E = \{e_{start}\} \cup \{e_{end}\} \cup E^{send} \cup E^{rec}$ comprises the start, end, message sending and message receiving events, respectively.
- The set of communicating tasks is $T^{com} = \{(t, e_{send}, c), (t, e_{rec}, c)\}$ where $t \in T,\ e_{send} \in E^{send},\ e_{rec} \in E^{rec},\ c \in C$.
- The set of operations $OP = \{F, M, XOR, ;\}$ comprises the parallel fork, parallel merge, exclusive choice and sequencing operations.
- C is a set of channels.

Business processlets and a business process are defined as follows:

- $t \in T \cup T^{com}$ is a processlet.
- If P and Q are processlets, then $P;Q$, $F(P,Q)M$ and $P(XOR)Q$ are processlets.
- If $P \in T \setminus T^{com}$ and e_{start}, e_{end} are the start and end events, then $e_{start}; P; e_{end}$ is a business process.

- Pools are business processes that satisfy the constraint: given pools P_1 and P_2, there is a task (t_1, e_{send}, c) in P_1 if and only if there is another task (t_2, e_{rec}, c) in P_2 such that a message can be sent using a channel c.

3.2 Middle Layer

The middle layer of the graphical model constructs potential attacks from evidence provided by intrusion detection system alerts and system logs. The objective is to map attack scenarios to missions that are modeled as business process diagrams. Because only attack scenarios substantiated using the available evidence are considered, the attack paths that are created do not include all the possible vulnerabilities and attack paths. In some cases, all the evidence may not be available in intrusion detection alerts and system logs; these situations are addressed by the lower layer.

Attack scenarios are reconstructed using a forensic analysis tool developed in previous work [4]. The tool uses rules to create directed graphs by correlating the available items of evidence. Because rules are used to create the graphs, they are referred to as logical evidence graphs (LEGs) [4].

Definition 2. (Logical Evidence Graph): A logical evidence graph is a six-tuple (N_r, N_f, N_c, E, L, G) where N_f, N_r and N_c are disjoint sets of nodes comprising fact, rule and consequence fact nodes, respectively. $E \subseteq ((N_f \cup N_c) \times N_r) \cup (N_r \times N_c)$ is the evidence, L is a mapping from nodes to labels and $G \subseteq N_c$ is a set of observed attack events. Every rule node has one or more fact nodes or consequence fact nodes from prior attack steps as its parents and a consequence fact node as its only child. Node labels consist of instantiations of rules or sets of predicates specified as follows:

1. A node in N_f is an instantiation of predicates that codifies system states, including access privileges, network topology and known vulnerabilities associated with host computers. The following predicates are used:
   - hasAccount(_principal,_host,_account), canAccessFile(_host, _user,_access,_path) and other predicates model access privileges.
   - attackerLocated(_host) and hacl(_src,_dst,_prot,_port) model network topology, including attacker location and network reachability information.

- vulExists(_host,_vulID,_program) and vulProperty(_vulID,_range,_consequence) model node vulnerabilities.

2. A node in N_r describes a single rule of the form $p \leftarrow p_1 \wedge p_2 \cdots \wedge p_n$. The rule head p is an instantiation of a predicate from N_c, which is the child node of N_r in the logical evidence graph. The rule body comprises p_i $(i = 1..n)$, which are predicate instantiations of N_f from the current attack step and N_c from one or more prior attack steps that comprise the parent nodes of N_r.

3. A node in N_c represents the predicate that codifies the post-attack state as the consequence of an attack step. The two predicates execCode(_host,_user) and netAccess(_machine,_protocol,_port) are used to model the attacker's capability after an attack step. Valid instantiations of these predicates after an attack update valid instantiations of the three predicates listed in item 1 above.

3.3 Lower Layer

The lower layer of the model uses instances of interactions between services and the execution environment to obtain evidence that is not provided by intrusion detection system alerts and system logs. In such situations, the interaction instances are obtained from system call logs. This is done because it is assumed that the missing evidence is due to the use of anti-forensic techniques, the limitations of forensic tools and/or the execution of zero-day attacks. Because there are many system calls, only the calls highlighted in [14] are considered; these calls are listed in the third column of Table 1. The first column of the table presents abstractions of the system calls. A process that makes system calls creates dependencies between itself and other processes, files or sockets for network connections. The dependencies are modeled as object dependency graphs (ODGs).

Definition 3. (Object Dependency Graph): The reflexive transitive closure of $\rightarrow$ defined in Table 1 is an object dependency graph. An object dependency graph is a three-tuple (V_O, V_E, D) where V_O is the set of vertices comprising objects (processes P, files F and sockets S), V_E is the set of textual descriptions of events (second column of Table 1) and D is the set of dependency edges listed in the first column of Table 1.

3.4 Mappings

The left-hand and right-hand portions of Figure 1 show the system resource mapping and graph mapping, respectively. The resource mapping

Table 1. System call dependencies.

Dependency	Event Description	Unix System Calls
process → file	process modifies file	`write`, `pwrite64`, `rename`, `mkdir`, `linkat`, `link`, `symlinkat`, etc
file → process	process reads file	`stat64`, `lstat6e`, `fsat64`, `open`, `read`, `pread64`, `execve`, etc.
process ↔ file	process uses/ modifies file	`open`, `rename`, `mount`, `mmap2`, `mprotect`, etc.
process1 → process 2	process1 creates/ terminates process2	`vfork`, `fork`, `kill`, etc.
process → socket	process writes socket	`write`, `pwrite64`, etc.
socket → process	process checks/ reads socket	`fstat64`, `read`, `pread64`, etc.
process ↔ socket	process reads/writes/ checks socket	`mount`, `connect`, `accept`, `bind`, `sendto`, `send`, `sendmsg`, etc.
socket ↔ socket	process reads/ writes socket	`connect`, `accept`, `sendto`, `sendmsg`, `recvfrom`, `recvmsg`

obtained from the infrastructure configuration and software deployment is used to map graphs. This is accomplished by mapping the attacked services in the corresponding vertices of business process diagrams, logical evidence graphs and object dependency graphs so that the source graphs can be mapped to the destination graphs. A logical evidence graph is easily mapped to a business process diagram by matching the attacked services to the corresponding tasks supported by the services. An object dependency graph is mapped to a logical evidence graph using depth-first search as specified in Algorithm 1.

In Algorithm 1, all the object nodes in an object dependency graph are initially marked as not been checked by using the color WHITE as shown in the for-loop in Lines 1–3. Then, for each unchecked object node V_O (Lines 4–5), the algorithm repeatedly calls function Find(V_O,LEG). The function call is on Line 10 and the function itself is located at Lines 28–41.

The function finds the matching post-attack status node in the logical evidence graph by checking if the attacked service in the logical evidence graph is the same as the attacked service in the object dependency graph.

Algorithm 1: Mapping an OEG to a LEG.

Input: ODG = (V, V_E, D) and LEG = (N_r, N_f, N_c, E, L, G).
Output: LEG integrated with attack paths from the ODG.

```
1  for each node V_O in ODG do
2      color[V_O] ← WHITE
3  end
4  for  each node V_O in ODG do
5      if V_O == WHITE then
6          for each node N_c in LEG do
7              color[N_c] ← WHITE
8          end
9          //Search for the corresponding N_c1 in LEG
10         N_c1 = Find(V_O, LEG)
11         //If there is a matching N_c1
12         if N_c1 ≠ ∅ then
13             color[V_O] ← BLACK
14             //Check if object parent matches the corresponding N_c1 parent
15             N_c2 = Find(parent(V_O), LEG)
16             //If there is no matching parent, add the missing attack step from
               ODG to LEG
17             if N_c2 ≠ parent(N_C1) then
18                 LEG ← Flow(N_c1, V_E); LEG ← Flow(V_E, N_c2)
19             end
20         end
21         else
22             //If there is no matching parent, add the new object to LEG
23             LEG ← V_O; color[V_O] = GRAY
24         end
25         V_O = child(V_O)
26     end
27 end
28 Function Find(V_O, LEG)
29 for each post-attack status N_c from LEG do
30     //Check if there is a matching N_c for V_O
31     if (N_c.service == V_O.service AND
32     color[N_c] == WHITE then
33         color[N_c] ← BLACK
34         return N_c
35     end
36     else
37         color[N_c] ← GRAY
38         N_c ← child post-attack status node of N_c
39     end
40 end
41 return ∅
```

If such a post-attack status node (say N_{c1}) is found (Line 12), then the algorithm checks if the attack step between node V_O and its parent parent(V_O) in the object dependency graph has a mapping attack step between node N_{c1} and its parent parent(N_{c1}) in the logical evidence graph (Line 15). If no matching attack step exists, then one is added to the logical evidence graph (Line 18). If no mapping post-attack status node N_{c1} exists in the logical evidence graph for node V_O in the object dependency graph (Line 21), then one is added to the logical evidence graph (Line 23), and the search continues (Line 25) until all the nodes in the object dependency graph are checked (i.e., colored).

3.5 Computing Mission Impacts

The mission impact of an attack is quantified using the [0,1] interval. This section provides details about the mission impact computations.

Computing Attack Impact Scores. In a logical evidence graph, the term $P(a)$ is used to denote the impact of an attack a on services deployed on a host computer. NIST's National Vulnerability Database – Common Vulnerability Scoring System lists vulnerabilities and assigns impact scores. If attack a is found in the database, then the corresponding score is used as the value for $P(a)$. If attack a is not found in the database, then expert knowledge is used to assign the impact score $P(a)$.

As proposed in [3], the cumulative impact score of attacks on the same service is computed using the equation:

$$P(a) = P(a_1) \cup P(a_2) \tag{1}$$

where a_1 and a_2 are two different attacks on the same service and:

$$P(a_1) \cup P(a_2) = P(a_1) + P(a_2) - P(a_1) \times P(a_2) \tag{2}$$

Assigning Weights to Tasks and Missions. The weight of the mission impact of an attack on a task is also quantified using the [0,1] interval. The higher the weight, the greater the importance of the task to the mission of a business process.

Computing Mission Attack Impacts. In this step, a logical evidence graph is mapped to the corresponding business process diagram and the mission impact of attacks $I(T)$ on a task T is computed using the following equation:

$$I(T) = weight \times P(T) \tag{3}$$

where $P(T)$ is the impact of attacks on task T in the business process diagram.

Depending on the mapping relationship from the attacked service(s) (represented by a, a_1, a_2) in a logical evidence graph to a task (represented by T) in a business process diagram, $P(T)$ is computed using the following equations for a one-to-one mapping relationship and a many-to-one mapping relationship, respectively:

$$P(T) = P(a) \quad (4)$$

$$P(T) = P(a_1) \cup P(a_2) \quad (5)$$

Computing the Cumulative Mission Impact. In some cases, the cumulative impact of attacks on the final mission is required to estimate the overall damage. The cumulative impact of attacks on the final mission C is computed in the following ways depending on the relationships existing between the tasks comprising a business process:

$$C(M) = \text{Max}\{I(T_1), I(T_2), \ldots, I(T_n)\} \quad (6)$$

$$C(M) = \text{Max}\{I(T_1), I(T_2), I(T_4) \ldots, I(T_n)\} \quad (7)$$

$$C(M) = \text{Max}\{I(T_1), I(T_3), I(T_4) \ldots, I(T_n)\} \quad (8)$$

$$C(M) = \text{Max}\{C(M_{before}), I(T_2')\} \quad (9)$$

- $C(M)$ is computed using Equation (6) when the tasks $T_1, T_2, \ldots, T_n$ comprising the final mission M have sequential relationships with each other or only some tasks among all the sequential tasks (e.g., T_2 and T_3) have parallel fork relationships with the predecessor task (T_1) and parallel merge relationships with the successor task (T_4).

- $C(M)$ is computed using Equations (7) or (8) when the tasks $T_1, T_2, \ldots, T_n$ comprising the final mission M include tasks (e.g., T_2 and T_3) that have exclusive decision relationships with the predecessor task (T_1) and successor task (T_4), and all the other tasks ($T_4, \ldots, T_n$) have sequential relationships with each other. Specifically, depending on whether task T_2 or task T_3 is chosen by the business process, either Equation (7) or Equation (8) is used to compute $C(M)$.

- Suppose tasks $T_1, T_2, \ldots, T_n$ comprise the final mission M in a pool and let $C(M_{before})$ denote the cumulative mission attack impact of M without any message passing from other pools. Then, Equation (9) is used to compute $C(M)$ if there is message passing

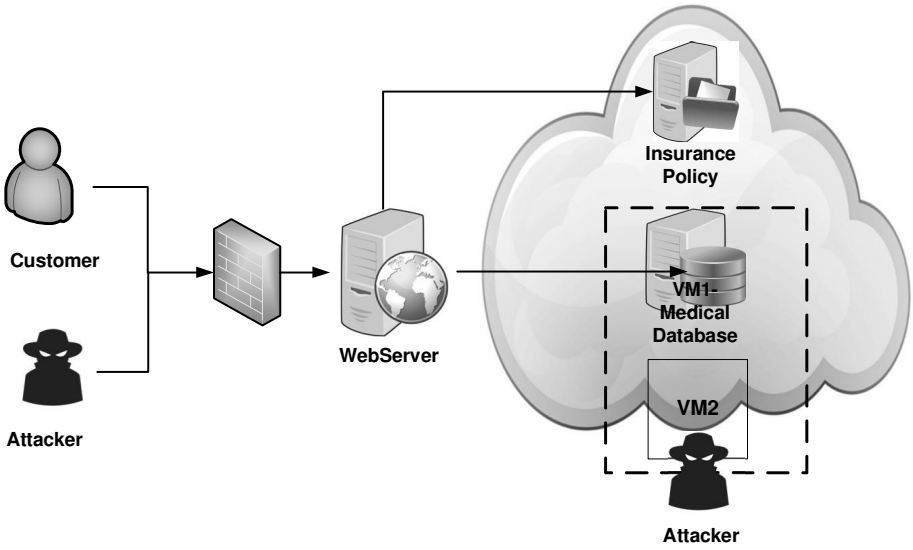


Figure 2. Experimental network.

between a task (T_2) in the pool and another task (T_2') in a different pool.

4. Case Study

This section uses a case study to demonstrate the utility of the proposed graphical model.

4.1 Experimental Network and Attacks

Figure 2 shows the experimental network used in the case study. The network was configured to manage customer medical records and health insurance policy files. The medical records and files were stored on two virtual machines (VM1 and VM2) in a private cloud deployed using OpenStack (Juno 2014.2.3) with a Xen hypervisor. OpenStack is a collection of Python-based software projects that manage access to pooled storage, computing and network resources that reside on one or more machines in a cloud system [10]. The projects include Neutron (networking), Nova (compute), Glance (image management), Swift (object storage), Cinder (block storage) and Keystone (authorization and authentication). OpenStack can be used to deploy a variety of cloud models, but is mostly deployed as an infrastructure-as-a-service.

In the experimental network, authenticated users were able to access the file server to retrieve policy files using `ssh` and to query the medical records stored on the database server using MySQL queries via a web application.

It was assumed that the attacker's objectives were to steal customer medical records, prevent medical record availability and modify health insurance policies. In the experiment, a simulated attacker probed the deployed web and cloud services, and launched four attacks: (i) SQL injection attack; (ii) denial-of-service (DoS) attack; (iii) cross-VM side-channel attack; and (iv) social engineering attack.

SQL Injection Attack. The web application did not sanitize user inputs. The attacker was able to exploit this vulnerability via a SQL injection attack (CWE-89) in order to access customer medical records. The following query was employed:

- `Select * from profile where name = 'Alice'` and `(password = 'alice' or '1' = '1')`

where `profile` is the database name and `'1'= '1'` is the payload that enables the query to bypass the password check. The SQL injection query retrieved all the customer medical records.

Denial-of-Service Attack. According to NIST's National Vulnerability Database, CVE-2015-3241 vulnerability in OpenStack Nova (compute) versions 2015.1 through 2015.1.1, 2014.2.3 and earlier enables authenticated users to cause denial-of-service by resizing and deleting virtual machine instances; the process of resizing and deleting an instance is called instance migration. Because of CVE-2015-3241, the migration process does not terminate when an instance is deleted, enabling an authenticated user to bypass user quota enforcement and deplete the available disk space by repeatedly performing instance migration.

In the experiment, the attacker played the role of a malicious privileged infrastructure-as-a-service user and launched a denial-of-service attack on the database server by repeatedly resizing and deleting VM2 that resided in the same physical machine as the database server (VM1).

Cross-VM Side-Channel Attack. Side-channel attacks can be used to extract fine-grained information across virtual machines that reside in the same hypervisor [19]. In the experimental network, the cache side-channel attack [18] shown in Figure 3 was launched against VM1 and VM2 that ran on the same multi-core processor (Intel quad-core i7). The cache side-channel attack leveraged the transparent page sharing feature implemented in hypervisors such as VMware ESXi and Xen.

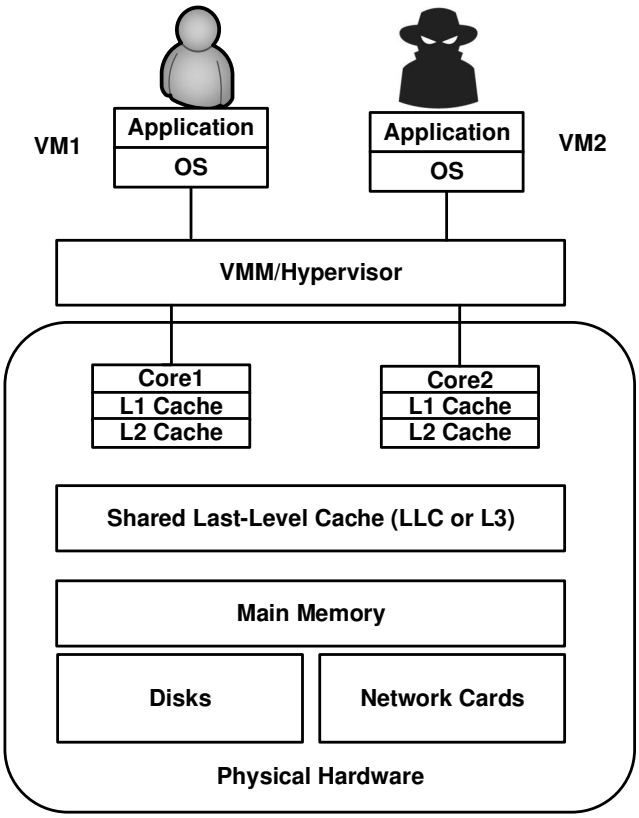


Figure 3. Cross-VM side-channel attack using a shared last-level cache.

This feature automatically identifies identical pages of virtual memory and consolidates them in a single physical memory page. In the experiment, the attacker exploited transparent page sharing to evict a specific memory line (i.e., memory unit in cache containing a fixed number of bytes) from all levels of the processor cache hierarchy, waited for a period of time and measured the time taken to load the data from the corresponding memory line. Because retrieving data from memory takes more time than retrieving it from cache levels closer to the core, the attacker was able to use the timing information and the data in the corresponding memory lines to obtain confidential information about the victim.

Since the attack leverages the implementation weakness in GnuPG 1.x before version 1.4.16 to obtain the information used to extract the private encryption key, GnuPG 1.4.12 was installed on VM1 (medical database server) while its copy and the spy program simultaneously ex-

ecuted on VM2. The copy of the attacked program executable was required because the attacker used the transparent page sharing feature, which coalesced memory pages from the victim's virtual machine VM1 and the attacker's virtual machine VM2.

Social Engineering Attack. The attacker executed a social engineering attack on the file server to obtain the administrator's credentials (username and password). The attacker then logged into the file server as the administrator and modified insurance policy files on the file server.

Evidence Capture. In order to capture evidence of attacks, the experimental network employed Wireshark for monitoring network traffic and Snort for intrusion detection; moreover, all the servers were configured to log user access. Additionally, to obtain evidence of attacks missed by Snort and the service logs, system calls by user processes in the two virtual machines were recorded.

4.2 Three Levels of Graphs

The network configuration, service deployment information and captured evidence were used to construct the three levels of graphs, which included a business process diagram, two logical evidence graphs and two object dependency graphs.

Business Process Diagram. Figure 4 shows the business process diagram of the experimental network. The network incorporated three pools: (i) Pool 1 (web interface); (ii) Pool 2 (public cloud service); and (iii) Pool 3 (infrastructure-as-a-service).

The business process in Pool 1 is the web interface for the clients (medical customers); it comprises the start and end events, two consecutive tasks (Enter Username/Password and Send Request) and an exclusive gateway for tasks (Review Policy File and Review Medical Records) depending on the client's request.

The business process in Pool 2 comprises the start and end events, one task (Check User Request) followed by an exclusive gateway that directs to two tasks (Request Policy Files and Request Customer Databases) depending on the message passed from task Send Request in Pool 1. In each decision task branch, there is an exclusive decision gateway (File Available and Data Available) followed by tasks that either send the requested data (Send Policy File or Send Customer Medical Records) via message passing back to the clients or reject the client request.

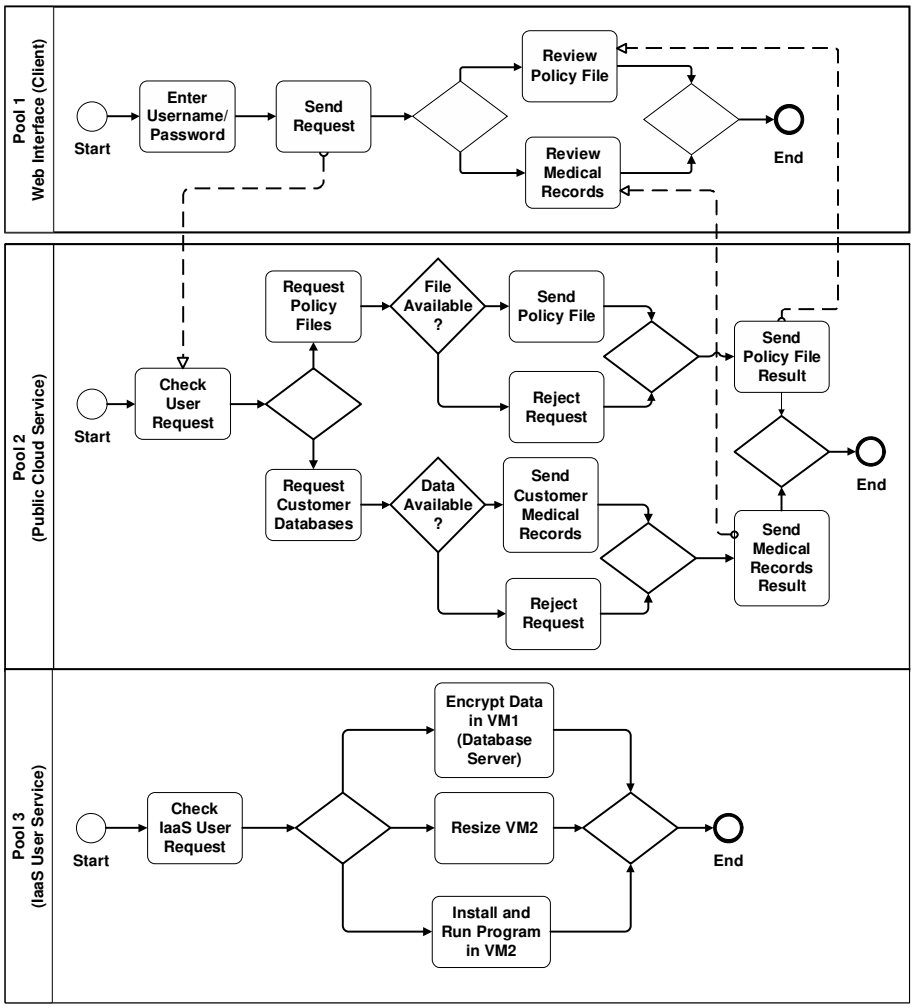


Figure 4. Business process diagram of the experimental network.

The business process in Pool 3 describes the user services; it comprises three tasks (Encrypt Data in VM1, Resize VM2 and Install and Run Program in VM2) that are the exclusive decisions of the Check IaaS User Request task. For each of the three business processes in the three pools, the last task(s) before the end event is taken to represent the entire business process mission(s).

Logical Evidence Graph. Table 2 shows evidence pertaining to the SQL injection attack, which includes the timestamps, machine IP addresses, Snort alerts and database access logs. Using the evidence, it

Table 2. SQL injection attack Snort alerts and database server log.

Timestamp	Machine	IP Address	Snort Alerts and Database Access Logs
	Attacker	129.174.124.122	
06/13/2017: 14:37:27	Web Server	129.174.124.184	SQL Injection Attack (CWE-89)
06/13/2017: 14:37:34	Database Server	129.174.124.35	Access from 129.174.124.184

```
2017-07-18 07:52:00.237 DEBUG oslo_concurrency.processutils
  [req-f79c7911-04ed-4a0c-adbe-0ae0a487c0f7 admin admin]
  Running cmd (subprocess): mv /opt/stack/data/nova/instances/
  bd1dac18-1ce2-44b5-93ee-967fec640ff3= /opt/stack/data/nova/
  instances/bd1dac18-1ce2-44b5-93ee-967fec640ff3_resize from
  (pid=41737)} execute /usr/local/lib/python2.7/dist-packages
  /oslo_concurrency/processutils.py:344
2017-07-18 07:52:00.253 DEBUG oslo_concurrency.processutils
  [req-f79c7911-04ed-4a0c-adbe-0ae0a487c0f7 admin admin] CMD
  "mv /opt/stack/data/nova/instances/bd1dac18-1ce2-44b5-93ee-
  967fec640ff3 /opt/stack/data/nova/instances/bd1dac18-1ce2-
  44b5-93ee-967fec640ff3_resize" returned: 0 in 0.016s from
  (pid=41737) execute /usr/local/lib/python2.7/dist-packages/
  oslo_concurrency/processutils.py:374
2017-07-18 07:52:00.254 DEBUG oslo_concurrency.processutils
  [req-f79c7911-04ed-4a0c-adbe-0ae0a487c0f7 admin admin]
  Running cmd (subprocess): mkdir p /opt/stack/data/nova/
  instances/bd1dac18-1ce2-44b5-93ee-967fec640ff3 from
  (pid=41737) execute /usr/local/lib/python2.7/dist-packages/
  oslo_concurrency/processutils.py:344
```

Figure 5. OpenStack Nova API call logs.

can be asserted that the attacker used a typical SQL injection with payload `'1' = '1'` to attack the customer database in the database server. Snort failed to capture evidence of the denial-of-service attack that exploited the vulnerability CVE-2015-3241 in OpenStack Nova services. The OpenStack application programing interface (API) logs provide information about user operations of running instances; some of the log entries are shown in the screenshot in Figure 5, where the second line in each entry is the user operation. This information leads to the conclusion that the user in VM2 (i.e., attacker in the experiment) kept resizing and deleting instances in VM2, which resided in the same physical ma-

```
/* initial attack location and final attack status */
attackerLocated(internet).
attackGoal(execCode(database,user)).
/* network access configuration */
hacl(internet,webServer,tcp,80).
hacl(webServer,database,tcp,3306).
/* configuration information of webServer */
vulExists(webServer,'directAccess',httpd).
vulProperty('directAccess',remoteExploit,privEscalation).
networkServiceInfo(webServer,httpd,tcp,80,apache).
/* vulnerability of the web application */
vulExists(database,'CWE-89',httpd).
vulProperty('CWE-89',remoteExploit,privEscalation).
networkServiceInfo(database,httpd,tcp,3306,user).
```

Figure 6. Prolog predicates for the SQL injection attack.

```
/* initial attack status of being an iaas user and final
   attack status */
attackerLocated(iaas).
attackGoal(execCode(nova,admin)).
/* cloud configuration, "_" represents any protocol
   and port */
hacl(iaas,nova,_,_).
/* vulnerability in nova */
vulExists(nova,'CVE-2015-3241','REST').
vulProperty('CVE-2015-3241',remoteExploit,privEscalation).
networkServiceInfo(nova,'REST',http,_,admin).
```

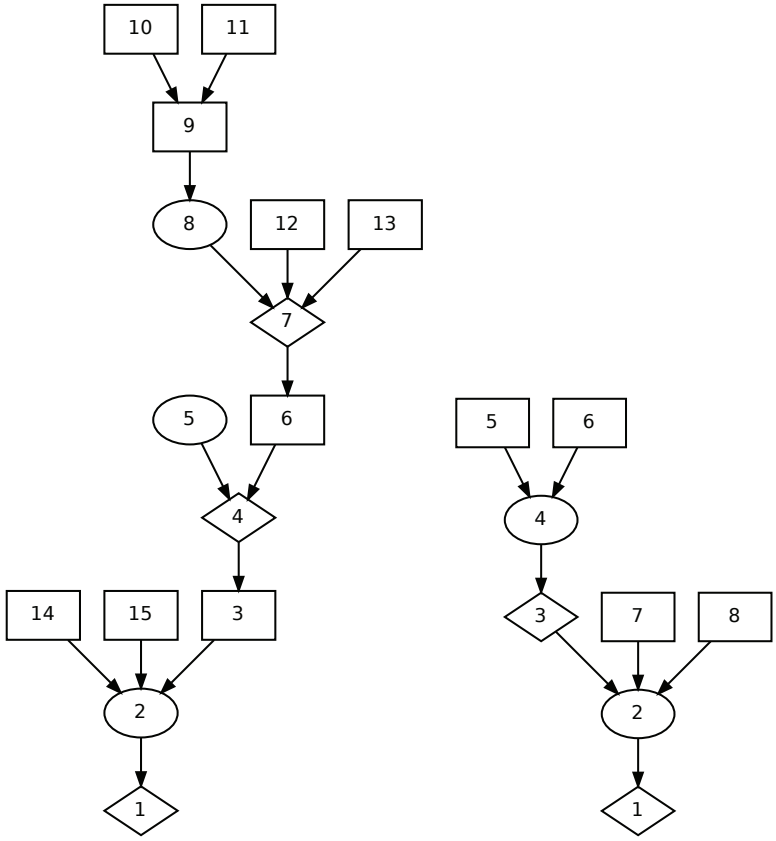
Figure 7. Prolog predicates for the denial-of-service attack.

chine as the database server (VM1) that launched the denial-of-service attack on the database server.

In order to use the forensic analysis tool, system configurations and the evidence related to the SQL injection and denial-of-service attacks were converted to the Prolog predicates shown in Figures 6 and 7.

Figures 8(a) and 8(b) show the output logical evidence graphs produced by the tool; Tables 3 and 4 provide descriptions of the nodes in the two graphs. The two logical evidence graphs are not grouped together due to distinct locations and privileges of the attacker.

Consider the attack step $3, 7, 8 \rightarrow 2 \rightarrow 1$ in Figure 8(b). Nodes 7 and 8 in the attack step model the pre-attack configurations and vulnerabilities. The consequence fact (Node 1) shows that post-attack evidence is derived by applying a rule (Node 2) to the existing system facts (Nodes 7 and 8) and the consequence fact from a prior step (Node 3).



(a) SQL injection attack on database. (b) DoS attack on database server.

Figure 8. SQL injection attack and DoS database attack LEGs.

```
execve("/home/flush-reload/build_gpg/gnupg-1.4.12/bin/gpg",
  ["/home/flush-reload/buil"..., "- -yes", "- -sign",
  ‘‘message.txt"], [/* 18 vars */]) = 0
...
```

Figure 9. Filtered system calls for the side-channel attack from VM1.

Object Dependency Graph. Due to the lack of intrusion detection system alerts and system logs for the cross-VM side-channel and social engineering attacks, it was not possible to reconstruct the two attack scenarios in the form of logical evidence graphs. Therefore, the system calls captured during the attacks were examined and the method presented above was used to construct object dependency graphs for a forensic analysis.

Table 3. Descriptions of the nodes in Figure 8(a).

Node	Notation
1	execCode(database,_)
2	RULE 2 (remote exploit of a server program)
3	netAccess(database,tcp,3306)
4	RULE 5 (multi-hop access)
5	hacl(webServer,database,tcp,3306)
6	execCode(webServer,apache)
7	RULE 2 (remote exploit of a server program)
8	netAccess(webServer,tcp,80)
9	RULE 6 (direct network access)
10	hacl(internet,webServer,tcp,80)
11	attackerLocated(internet)
12	networkServiceInfo(webServer,httpd,tcp,80,apache)
13	vulExists(webServer,'directAccess',httpd, remoteExploit,privEscalation)
14	networkServiceInfo(database,httpd,tcp,3306,_)
15	vulExists(database,'CWE-89',httpd,remoteExploit,privEscalation)

Table 4. Descriptions of the nodes in Figure 8(b).

Node	Notation
1	execCode(nova,admin)
2	RULE 2 (remote exploit of a server program)
3	netAccess(nova,http,_)
4	RULE 6 (direct network access)
5	hacl(cloud,nova,http,_)
6	attackerLocated(cloud)
7	networkServiceInfo(nova,'REST',http,_,admin)
8	vulExists(nova,'CVE-2015-3241','REST',remoteExploit,privEscalation)

Figure 9 shows some system calls captured from VM1 (database server) and VM2 (attacker) during the cross-VM side-channel attack. The system calls show that file `message.txt` in VM1 was encrypted using GnuPG 1.4.12.

The system calls in Figure 10 show that a probe program `bin/probe` in VM2 used `mmap2` to force the system to share memory addresses with the attacker's probing process. This enabled the probing process to read data from the shared memory addresses and write the data to an output file for malicious purposes.

```
execve("bin/probe", ["bin/probe", "/home/flush-reload/buil"...,
    "docs/addr/osx.txt", "out.txt", "200"], [/* 18 vars */]) = 0
...
fstat(3, {st_mode=S_IFREG\0644, st_size=101240,...}) = 0
mmap(NULL, 2206376, PROT_READ|PROT_EXEC, MAP_PRIVATE|
    MAP_DENYWRITE, 3, 0) = 0x7f80b73ec000
read(4, "4\n0x00000000000967c7\n0x000000000" ..., 4096) = 78
write(1, "Probing 4 addresses:\n", 21) = 21
write(1, "0x967c7\n", 8) = 8
write(1, "0x95f5d\n", 8) = 8
write(1, "0x97225\n", 8) = 8
write(1, "0x9757f\n", 8) = 8
...
write(1, "Started spying\n", 15) = 15
...
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_ANONYMOUS, -1, 0) = 0x7f5e24ab7000
write(5, "0 0 47064\n0 1 6290\n0 2 6246\n0 3"..., 4096) = 4096
write(5, "3 1 279\n113 2 229\n113 3 254\n114" ..., 4096) = 4096
write(5, "9\n215 3 257\n216 0 239\n216 1 279\n" ..., 4096) = 4096
...
```

Figure 10. Filtered system calls for the side-channel attack from VM2.

Figure 11 shows some of the system calls captured from the file server. The read/write system call trace shows that `test.txt` in the file server was modified. Because the corresponding `sshd` log in the file server recorded user accesses, it was easy to conclude that the attacker stole the file server administrator's credentials to modify the insurance policy file (the `sshd` log is omitted for reasons of space).

The system calls and dependency rules listed in Table 1 were used to filter the important system calls. Two object dependency graphs were subsequently constructed. One object dependency graph showed that the attacker in VM2 read the shared cache between VM1 and VM2. The other graph showed that the attacker used the file server administrator's credentials from the Internet to modify an insurance policy file in the file server. The two object dependency graphs were then mapped to the logical evidence graphs in Figure 8.

The resulting integrated logical evidence graphs revealed that:

- The attacker launched two attacks from the Internet. One attack involved the use of the stolen credentials to modify an insurance policy file and the other, a SQL injection attack, involved the theft of customer medical records.

```
write(9, "v", 1) = 1
read(11, "v", 16384) = 1
write(3, "\0\0\0\20\331\255\275\264c\2173)z2j\32\255n\2007d
    \366m\21\316\2648\240\207\31\211" ..., 36) = 36
read(3, "\0\0\0\20\240\253\341\227\321xU\305\347\226\246\361/316
    \242S=\30\341QT\231\n\343\314\343\307\f\361" ..., 16384) = 36
write(9, "i", 1) = 1
read(11, "i", 16384) = 1
write(3, "\0\0\0\20\177\352\313\332\373yjM\34161\230\215\10\220p
    \252g\375\365\1\f\335\361\r\273\374\357" ..., 36) = 36
read(3, "\0\0\0\20\27\334?\201x\300\16\356\346\0379\32\220\372
    \366\4\v\1=\347\263\311\250k\353" ..., 16384) = 36
write(9, "", 1) = 1
read(11, "", 16384) = 1
write(3, "\0\0\0\20i\321\344\220\313\322\254S\252o\201\225;6v
    \243\205\10gs\253\237\325\375\332v" ..., 36) = 36
read(3, "\0\0\0\20\5\27k;\254\301\24\n\ZN\267\260\336\323\323
    \32\345\2b\226-\271}[B\21" ..., 16384) = 36
write(9, "t", 1) = 1
read(11, "t", 16384) = 1
read(3, "\0\0\0\20\325\261\7\254\211(\201\331\272\344[\355\200u4
    \357G\347\232\276:\201\376\342\202\201" ..., 16384) = 36
write(3, "\0\0\0\20\320\254\#\312\211_\3022\n\227\16I\372\202
    \347\37\252T\257\220\210E\343\222\342\24S" ..., 36) = 36
write(9, "e", 1) = 1
read(11, "e", 16384) = 1
...
write(9, "t", 1) = 1
read(11, "st.txt", 16384) = 7
...
```

Figure 11. Filtered system calls for the file modification from the file server.

- The attacker launched two other attacks as an infrastructure-as-a-service user. One was a denial-of-service attack and the other was a cross-VM side-channel attack on the database server.

Table 5. CVSS attack impact scores.

Attack	CVE Entry	Symbol	Attack Impact
SQL Injection Attack	CWE-89	N_1	0.90
Denial-of-Service Attack	CVE-2015-3241	N_1'	0.69
Social Engineering Attack		N_s	0.50
Side-Channel Attack	CVE-2013-4576	N_{sc}	0.29

4.3 Computing Mission Impact

Table 5 lists the impact scores of the reconstructed attacks. The impact scores for CWE-89, CVE-2015-3241 and CVE-2013-4576 were

obtained from NIST's National Vulnerability Database – Common Vulnerability Scoring System. The impact score for the social engineering attack was based on expert knowledge. The impact scores, which were originally specified on a [0,10] scale, were converted to the [0,1] scale.

All the reconstructed attacks were associated with the corresponding tasks by mapping object dependency graphs to logical evidence graphs and, eventually, to the business process diagram as shown in Figure 4. Table 6 shows the mission impact scores that were computed using the graph mappings.

Finally, using the attack impact scores for all tasks listed in Table 6 and the business process diagram shown in Figure 4, the cumulative attack impacts on the final missions were computed. Table 7 presents the results of the computations.

4.4 Reducing Attack Risk

In complex infrastructures, enterprise missions rely on combinations of and connections between multiple services. Because each service is supported by software and hardware assets, which are usually the targets of attackers, a tool that relates the infrastructure assets to the final missions can help determine the impacts of cyber attacks on enterprise missions. By correlating the attacks on lower-level assets to the higher-level business process diagram and using mission impact scores of the attacks as listed in Table 5, the proposed graphical model enables the computation of the impacts of attacks on complex missions. This information is valuable to digital forensic investigators and infrastructure risk managers.

The experimental network can be used as an exemplar to demonstrate how the computed mission impacts can be used by managers to reduce the risks of attacks. The cumulative impact scores in Table 7 show that the attacks on the Review Medical Records mission have a higher impact because customer medical records could be stolen via a SQL injection attack. This suggests that input sanitization should be implemented to defeat SQL injection attacks.

Additionally, the attacks and their impact scores in Table 6 reveal that two attacks – the denial-of-service attack and the side-channel attack – exploited cloud service vulnerabilities. Therefore, the services should be moved to a stable cloud that implements countermeasures for attacks launched from the shared hypervisor or physical machine.

Finally, Table 6 shows that, although the mission impact on the health insurance policies stored on the file server might not be as bad as the SQL injection attack on the database server, its score of 0.5 cannot be

Table 6. Mission impact scores.

Pool	Task	Attack	Weight	Mission Impact
Pool 1	Check Username and Password	CWE-89 (N_1)	1.0	$I_1 = 1 \times P(N_1) = 1 \times 0.9 = 0.9$
Pool 2	Data Available	CVE-2015-3241 (N_1')	0.9	$I_2 = 0.9 \times P(N_1') = 0.9 \times 0.69 = 0.621$
Pool 2	Request Policy Files	Social Engineering (N_s)	1.0	$I_3 = 1 \times P(N_s) = 1 \times 0.5 = 0.5$
Pool 3	Encrypt Data in VM1	CVE-2013-4576 (N_{sc})	1.0	$I_4 = 1 \times P(N_{sc}) = 1 \times 0.29 = 0.29$

Table 7. Cumulative mission impact.

Pool	Mission	Mission Task Attack	Cumulative Mission Impact
Pool 1	Review Policy File	Social Engineering	$C = \text{Max}(I_3) = \text{Max}(0.5) = 0.5$
Pool 1	Review Medical Records	Check Username and Password; Data Available	$C = \text{Max}(I_1, I_2) = \text{Max}(0.9, 0.621) = 0.9$
Pool 3	Encrypt Data in VM1	Encrypt Data in VM1	$C = \text{Max}(I_4) = \text{Max}(0.29) = 0.29$

ignored. A simple attack countermeasure is to restrict file write/modify rights to administrators with local access.

5. Conclusions

The three-layer graphical model presented in this chapter is designed to support forensic investigations of attacks on computing infrastructures. It helps reconstruct attack scenarios based on evidence from attack logs and leverages system call sequences when the logs do not have adequate evidence to reconstruct attack steps. NIST's National Vulnerability Database – Common Vulnerability Scoring System scores are used to compute the mission impacts of attacks; attacks that are not included in the NIST database are handled using estimates provided by human experts. The case study involving an experimental network with cloud services demonstrates the utility of the graphical model and its ability to support forensic investigations and risk mitigation efforts.

Future research will investigate applications of the graphical model in advanced networking and cloud infrastructures. Additionally, research will focus on determining how the model can be engaged in a framework that would enable enterprises to measure and manage the security risks to their infrastructures.

References

[1] L. Herbert, Specification, Verification and Optimization of Business Processes: A Unified Framework, Ph.D. Dissertation, Department of Applied Mathematics and Computer Science, Technical University of Denmark, Kongens Lyngby, Denmark, 2014.

[2] S. Jajodia and S. Noel, Topological vulnerability analysis, in *Cyber Situational Awareness*, S. Jajodia, P. Liu, V. Swarup and C. Wang (Eds.), Springer, Boston, Massachusetts, pp. 139–154, 2010.

[3] C. Liu, A. Singhal and D. Wijesekera, Mapping evidence graphs to attack graphs, *Proceedings of the IEEE International Workshop on Information Forensics and Security*, pp. 121–126, 2012.

[4] C. Liu, A. Singhal and D. Wijesekera, A logic-based network forensic model for evidence analysis, in *Advances in Digital Forensics XI*, G. Peterson and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 129–145, 2015.

[5] C. Liu, A. Singhal and D. Wijesekara, A probabilistic network forensic model for evidence analysis, in *Advances in Digital Forensics XII*, G. Peterson and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 189–210, 2016.

[6] P. Mell and T. Grance, NIST Definition of Cloud Computing, NIST Special Publication 800-145, National Institute of Standards and Technology, Gaithersburg, Maryland, 2011.

[7] S. Musman and A. Temin, A cyber mission impact assessment tool, *Proceedings of the IEEE International Symposium on Technologies for Homeland Security*, 2015.

[8] National Institute of Standards and Technology, National Vulnerability Database, Gaithersburg, Maryland (`nvd.nist.gov/vuln-metrics/cvss`), 2018.

[9] S. Noel, J. Ludwig, P. Jain, D. Johnson, R. Thomas, J. McFarland, B. King, S. Webster and B. Tello, Analyzing mission impacts of cyber actions (AMICA), *Proceedings of the NATO IST-128 Workshop: Assessing Mission Impact of Cyberattacks*, pp. 80–86, 2015.

[10] OpenStack Foundation, Software, Austin, Texas (`www.openstack.org/software`), 2018.

[11] X. Ou, S. Govindavajhala and A. Appel, MulVAL: A logic-based network security analyzer, *Proceedings of the Fourteenth USENIX Security Symposium*, 2005.

[12] K. Ruan, J. Carthy, T. Kechadi and M. Crosbie, Cloud forensics, in *Advances in Digital Forensics V*, G. Peterson and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 35–46, 2011.

[13] M. Saudi, An Overview of a Disk Imaging Tool in Computer Forensics, InfoSec Reading Room, SANS Institute, Bethesda, Maryland, 2001.

[14] X. Sun, J. Dai, P. Liu, A. Singhal and J. Yen, Towards probabilistic identification of zero-day attack paths, *Proceedings of the IEEE Conference on Communications and Network Security*, pp. 64–72, 2016.

[15] X. Sun, A. Singhal and P. Liu, Towards actionable mission impact assessment in the context of cloud computing, in *Data and Applications Security and Privacy XXXI*, G. Livraga and S. Zhu (Eds), Springer International, Cham, Switzerland, pp. 259–274, 2017.

[16] Y. Sun, T. Wu, X. Liu and M. Obaidat, Multilayered impact evaluation model for attacking missions, *IEEE Systems Journal*, vol. 10(4), pp. 1304–1315, 2016.

[17] W. Wang and T. Daniels, A graph based approach toward network forensic analysis, *ACM Transactions on Information and Systems Security*, vol. 12(1), article no. 4, 2008.

[18] Y. Yarom and K. Falkner, FLUSH+RELOAD: A high resolution, low noise, L3 cache side-channel attack, *Proceedings of the Twenty-Third USENIX Security Symposium*, pp. 719–732, 2014.

[19] Y. Zhang, A. Juels, M. Reiter and T. Ristenpart, Cross-VM side channels and their use to extract private keys, *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 305–316, 2012.