Who Touched My Mission: Towards Probabilistic Mission Impact Assessment

Xiaoyan Sun
Pennsylvania State University
University Park, PA 16802, USA
xzs5052@ist.psu.edu

Anoop Singhal
National Institute of Standards and Technology
Gaithersburg, MD 20899, USA
anoop.singhal@nist.gov

Peng Liu
Pennsylvania State University
University Park, PA 16802, USA
pliu@ist.psu.edu

ABSTRACT

Cyber attacks inevitably generate impacts towards relevant missions. However, concrete methods to accurately evaluate such impacts are rare. In this paper, we propose a probabilistic approach based on Bayesian networks for quantitative mission impact assessment. A System Object Dependency Graph (SODG) is first built to capture the intrusion propagation process at the low operating system level. On top of the SODG, a mission-task-asset (MTA) map can be established to associate the system objects with corresponding tasks and missions. Based on the MTA map, a Bayesian network can be constructed to leverage the collected intrusion evidence and infer the probabilities of tasks and missions being tainted. An example MTA-based BN is provided to show how our approach can enable effective quantitative mission impact assessment.

Categories and Subject Descriptors

K.6.m [**MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS**]: Miscellaneous

General Terms

Security

Keywords

Mission impact assessment; Bayesian network; System Object Dependency Graph

1. INTRODUCTION

Defending missions in cyber space from various attacks continues to be a challenge. An effective attack can lead to great loss in the confidentiality, integrity, or availability to the missions, and even cause some to abort in extreme cases [1]. When an attack happens, one major concern to the security administrators is how the attack could possibly impact related missions. Specifically, they may ask the questions such as 1) How likely is a mission affected? 2) To what extent is the mission influenced? Which tasks are already tainted, and which are untouched?

Continuous efforts have been made to construct high-level models that aid the mission impact analysis, but concrete methods that achieve accurate quantitative assessment are rare. Dai et al. [2] propose a Situation Knowledge Reference Model (SKRM) that enables mission damage and impact assessment. However, without rigidly specifying the cross-layer interconnections, SKRM lacks the capability of performing quantitative mission impact analysis. Jackobson [1] constructs an impact dependency graph (IDG) for mission situation assessment. Nevertheless, the paper doesn't specify detailed method for generating the dependencies in the IDG. The impact assessment provided by the IDG is not sufficiently precise.

In this paper, we propose a probabilistic approach based on Bayesian networks (BN) for mission impact assessment. Our approach is to 1) build a System Object Dependency Graph (SODG) so that the intrusion propagation process is captured at the system object level; 2) construct a Mission-Task-Asset (MTA) map to associate the missions and composing tasks with corresponding assets, which are namely the system objects such as processes, files, etc. The MTA map is naturally connected to the SODG through shared system objects; 3) establish a Bayesian network based on the MTA map and the SODG to leverage the collected intrusion evidence and infer the probabilities of interested events, such as a system object or a mission task being tainted.

The approach is proposed on the basis of the following supporting rationales. First, the SODG is a proper construct connecting the attack and the missions, as shown in Figure 1. From the attack side, an attack's impact towards the operating systems can be reflected on the SODG. System objects that are manipulated directly or indirectly by attackers have the possibility of being tainted. From the mission side, a mission is fulfilled through a sequence of operations towards system objects. These operations are caught by the SODG. As a result, the impact of an attack to the missions can be evaluated by leveraging the SODG as the intermediate bridge.

Second, the SODG is able to capture the intrusion propagation process, which is critical for correct mission impact assessment. An attack's impact towards a mission may not be explicit when they have no common associated assets. The attack-associated assets refer to the system ob-

SafeConfig'15, October 12, 2015, Denver, Colorado, USA.

DOI: http://dx.doi.org/10.1145/2809826.2809834.

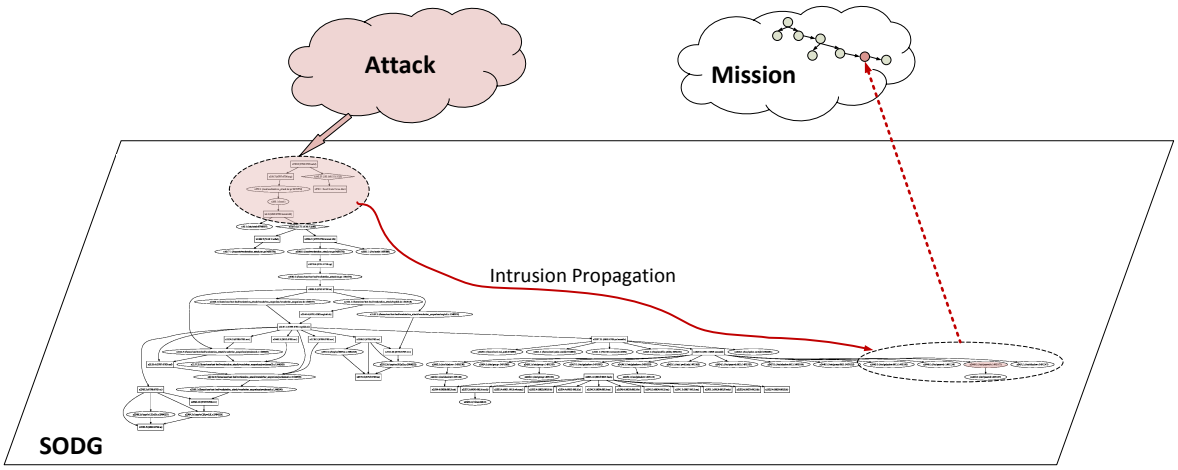


Figure 1: The SODG as the Construct between Attack and Mission [1]

jects that are directly related to the attack activities (e.g. a modified file in a Tripwire [3] alert), while the mission-associated assets refer to the system objects that are involved in the mission commitment. The mission-associated assets do not always share the same system objects with the attack-associated assets, but can still be affected by the latter through intrusion propagation. In this case, the SODG can be employed for tracking the intrusion propagation and assessing the missions that are indirectly affected by the attack-associated assets.

Third, a Bayesian network is able to leverage intrusion evidence to perform probabilistic inference towards interesting events. The evidence can be collected from a variety of information sources, including system logs, security sensors such as Snort [4] and Tcpdump [5], and even human experts.

The paper is organized as follows. Section 2 introduces the System Object Dependency Graph (SODG). Section 3 presents the main principles for establishing the mission-task-asset map. Section 4 briefly discusses the MTA-based Bayesian networks. Section 5 describes the related work. Section 6 concludes the whole paper.

2. THE SYSTEM OBJECT DEPENDENCY GRAPH

In essence, a mission can be decomposed to a set of tasks, which are then committed through a number of operating system operations via system calls, such as read, write, execve, fork, kill, etc. These system calls operate towards system objects like processes, files, and sockets. For instance, the system call *read* can read from a file and *fork* creates a copy of a process. An intrusion usually begins with one or more tainted system objects that are directly or indirectly manipulated by attackers. For example, an execution file containing a Trojan horse may have been installed on a host; a service may have been compromised with a rootkit program and started sending sensitive data back to the attackers' machine; some critical data that influences the control flow could have been corrupted so that the execution paths of a mission workflow can be changed. In subsequent system calls, these intrusion-originating system objects will interact with other innocent objects and get them tainted. This is an *intrusion propagation* process. In this way, the intrusion can propagate across a number of systems inside a network. Among all the system objects tainted via intrusion propagation, some could be the mission-associated ones so that the related tasks will get impacted as well.

[1]The SODG is used to show how the intrusion can propagate from the attack associated assets to the mission assocaited assets. Readers are not expected to understand the details inside the nodes of the SODG.

Table 1: System Call Dependency Rules

System calls	Dependency
write, pwrite64, rename, mkdir, fchmod, chmod, fchownat, etc.	process→file
stat64, read, pread64, execve, etc.	file→process
vfork, fork, kill, etc.	process→process
write, pwrite64, send, sendmsg, etc.	process→socket
read, pread64,recv, recvmsg, etc.	socket→process
sendmsg, recvmsg, etc.	socket→socket

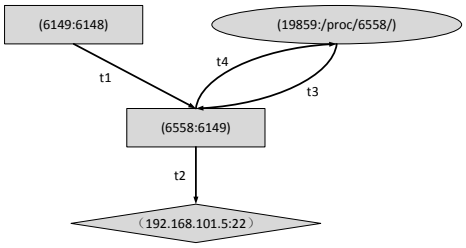
Given the system call log, a *System Object Dependency Graph (SODG)* can be constructed to capture the intrusion propagation process [8]. Each system call is first parsed into three elements: a source object, a sink object, and a dependency relation between them. This paper applies similar rules, shown in Table 1, as in [6–8] for system call parsing. When constructing the SODG, the parsed objects become nodes and the dependency relations become edges. For example, a *read* system call can be parsed into a process object p, a file object f, and a dependency relation $f \rightarrow p$, meaning that p depends on f.

Fig. 2b shows an example SODG built from a simplified system call log in Fig. 2a. Processes, files, and sockets are represented with rectangles, ellipses, and diamonds respectively. A process is often uniquely identified by the process PID *pid* and the parent process PID *ppid*, and thus can be denoted with a tuple (*pid*:*ppid*). Similarly, a file and a socket can be denoted with tuple (*inode*:*path*) and (*addr*:*port*).

The SODG construction process for Figure 2b is as follows. First, the system call *clone* is parsed into a dependency (6149 : 6148)→(6558 : 6149). The dependency be-

```
syscall:clone time:t1 pid:6149 ppid:6148 pcmd:bash
cpid:6558  cppid:6149 cpcmd:bash
syscall:write time:t2 pid:6558 ppid:6149 pcmd:sshd
ftype:SOCK addr:192.168.101.5 port:22
syscall:read  time:t3 pid:6558 ppid:6149 pcmd:mount
ftype:REG  path:/proc/6558/ inode:19859
syscall:write time:t4 pid:6558 ppid:6149 pcmd:sshd
ftype:REG  path:/proc/6558/ inode:19859
```

(a) simplified system call log



(b) SODG

Figure 2: An example SODG built from the simplified system call log

comes an edge between the two processes. Second, the system call *write* forms a dependency between a process and a socket: $(6558 : 6149) \rightarrow (192.168.101.5 : 22)$. The dependency becomes an edge between the process and the socket. Third, the system call *read* indicates that the process then reads a file, and thus creates a dependency $(19859 : /proc/6558/) \rightarrow (6558 : 6149)$. Finally, the process writes back to the same file, and forms a dependency $(19859 : /proc/6558/) \leftarrow (6558 : 6149)$.

After the SODG is constructed, forward and backward tracking can be performed to identify the potentially tainted objects. Since an attack can often cause security sensors to raise alerts, the system objects involved in these alerts can be used as the trigger points that start the tracking process. For example, if Tripwire raises an alert that a file is modified abnormally, then the file can be used as a trigger point. On the SODG, the file is marked as tainted. Starting from this file, forward and backward tracking can be performed to generate an intrusion propagation path [8]. The objects on this path are very likely to be tainted.

3. MISSION-TASK-ASSET MAP

Constructing Mission-Task-Asset (MTA) map is to relate the system objects with the tasks and missions. An intuitive solution is to decompose the missions into tasks, and further associate the tasks with system objects. However, this top-down decomposing approach requires the prior knowledge of a mission workflow. In cases when attackers are able to insert malicious tasks into the workflow, these inserted tasks could be missed by the MTA map.

In this paper, we propose a *bottom-up extraction* approach that extracts the tasks from the SODG, and then relates the tasks with specific missions, as shown in Figure 3. Since the SODG captures what actually happens in the network, extraction from the SODG accurately reflects which tasks are actually committed. Considering the manageable number of missions and tasks an enterprise network could deal with, relating tasks with missions is not a real issue. The key difficulty lies in how to extract tasks from the SODG due to its daunting size. However, the extraction is ensured to be feasible by the following principles.

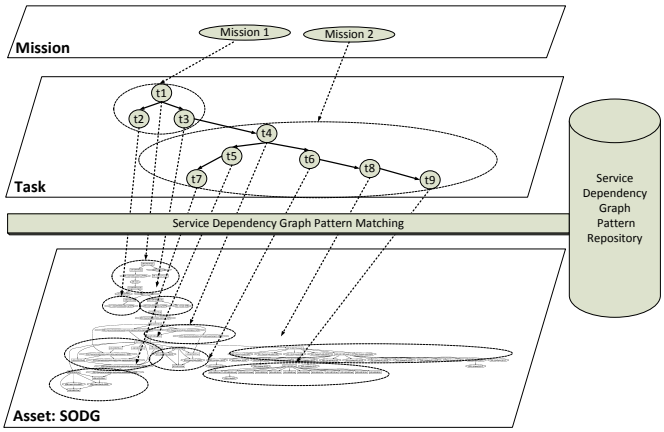


Figure 3: Mission-Task-Asset Map [2]

First, a mission task can be viewed as an instantiation of several services that have dependency relations. In enterprise networks, the normal function of a service may depend on one or more other services. These services and applications often interact and work together to accomplish specific tasks. For example, a user's login request requires web service from a web server, which further relies on au authentication service to verify the user's legitimacy. The authentication will then depend on the database service to access the users' account information. In this example, a single task "user login" can be viewed as the instantiation of combined web service, authentication service, and database service. Therefore, if such dependency relations among services can be discovered and represented with specific graphs, then a task can be viewed as the instantiation of a service dependency graph.

Second, through service discovery, the service dependency graphs (SDGs) can be established at the system object level. Service discovery has been studied intensively in previous work [9–12]. Dai [13] proposed to infer the service dependency through identifying OS-level causal paths. Therefore, the service dependencies can be represented with OS-level dependency graphs, such as the SODGs. Each service dependency graph has a pattern that can be used to identify the corresponding SDG. The patterns could be defined from the perspective of both text and graph-topology. For example, a file node with name *config* and an out degree of n can be one feature for a specific pattern, indicating that file *config* is accessed n times. Since servers in an enterprise network often fulfill routine responsibilities, the common patterns can be extracted to form an SDG pattern repository.

[2] Again, readers are not expected to understand the details inside the nodes of the SODG.

Third, the system assets can be linked to tasks automatically by matching the SODG against the SDG patterns. Although the SODG is usually not human-readable, it can be annotated with specific SDGs through pattern matching. For example, if the pattern for combined web service, authentication service, and database service appears in the SODG for several times, then as the instantiations of these services, several "user login" tasks can be linked to the system objects involved in these patterns.

4. BAYESIAN NETWORKS

To perform probabilistic mission impact assessment, the Bayesian networks can be constructed based on the established MTA maps. The Bayesian network is a type of Directed Acyclic Graph that can be used to model the cause and effect relations. In a BN, the nodes represent the variables of interest, and the edges represent the causality relations between nodes. The strength of such causality relations can be specified with conditional probability tables (CPT). When evidence is provided, a properly constructed BN can infer the probabilities of interesting variables.

In this paper, we propose to construct an MTA-based BN, whose input is the intrusion evidence collected from various security sensors, and output is the probabilities of interesting security events, such as a system object or a task being tainted. The graphical feature of MTA enables and facilitates the construction of MTA-based BN. With CPT tables specified and the evidence incorporated, the MTA-based BN is able to infer the probabilities of tasks and missions being tainted, and thus evaluate the impact of attacks towards interesting missions.

To build the MTA-based BN, the dependency relations existing in the MTA map need to be well modeled. Each MTA map implies certain dependency relations among the missions, tasks, and system objects. Such dependency relations can be represented with certain mission dependency graphs by interpreting the MTA maps. In the mission dependency graph, the status of a mission depends on the status of the composing tasks, while the status of a task depends on the status of the relevant system objects. We provide two example mission dependency graphs based on the same MTA map to illustrate how the dependency relations can be interpreted.

Figure 4 is an example of benign mission dependency graph by interpreting an MTA map. In this graph, a mission is composed of several tasks. For each mission to be benign, all of its composing tasks should be benign. In addition, all the tasks should be committed in the correct sequence. Similarly, each task is also composed of several system level operations. To ensure the task is benign, the related system objects should be benign and the operations should be performed in the right sequence. Therefore, all of the parent nodes have the "AND" relation for the child node to be true. In Figure 4, Node 5 "Task 1 is benign" should have 4 preconditions satisfied in order to be true: Node 1, F1 is benign; Node 2, P1 is benign; Node 3, F2 is benign; Node 4, "Process P1 reads File F1" happens before "Process P1 writes File F2", meaning that the read operation is executed before the write operation. In this example, in order for Node 5 to become true, all the relevant system objects are benign and all the system operations are performed in the right sequence. The relationship between these conditions (Node 1 to 4) is "AND".

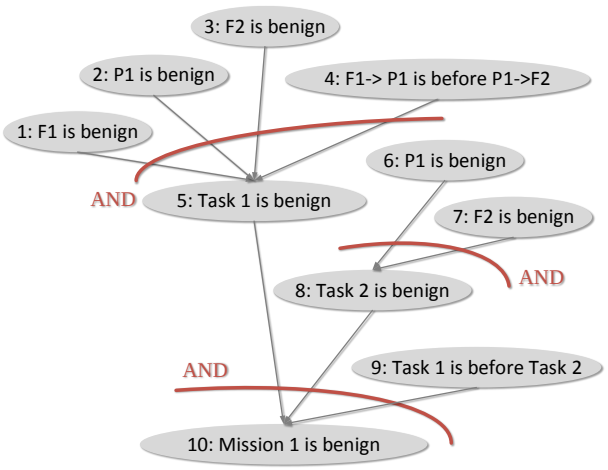


Figure 4: An Example of Benign Mission Dependency Graph

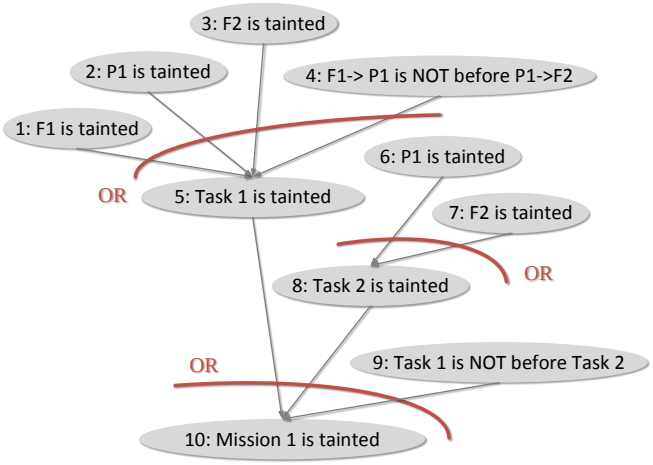


Figure 5: An Example of Tainted Mission Dependency Graph

Figure 5 is an example of a tainted mission dependency graph by interpreting the same MTA map as in Figure 4. In this graph, if any of the system objects are tainted or the system operations are not performed in the right order, the associated task can be marked as tainted. Similarly, if any of the tasks are tainted or not committed in the correct sequence, the associated mission is tainted. Therefore, all the parent nodes have the "OR" relation for the child node to be true, meaning any of the preconditions being true could cause the post-condition effective. For example, even if only F1 in Node 1 is tainted while F2 and P1 are still benign, Task 1 will get tainted, which will further impacts Mission 1.

To model the above "AND" and "OR" relations, a MTA-based BN can be constructed as shown in Figure 6. Instead of specifying the taint status of objects, tasks, and missions

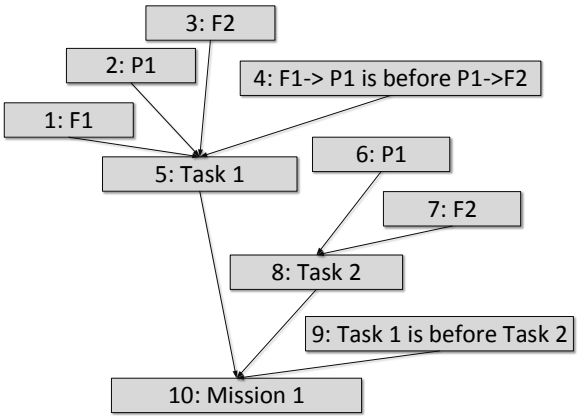


Figure 6: An Example of MTA-based BN

in the nodes directly, the MTA-based BN specify the states in the CPT tables. For example, the CPT table for Mission 1 in Figure 6 is shown in Table 2. In this table, Mission 1, Task 1, and Task 2 have possible states of "tainted" and "not tainted". The operation sequence "Task 1 is before Task 2" in Node 9 has the states of "true" and "false". Other potential states, such as "clear but in danger", or "not sure", etc, could also be assigned for system objects depending on specific situations.

In addition, the numbers in Table 2 modeled the "AND" and "OR" relations. For example, to get "mission 1 = not tainted" the probability of 1, all the three conditions "Task 1 is tainted", "Task 2 is tainted", and "Task 1 is before Task 2" have to be false. As long as any of these three conditions are true, the probability for "mission 1 = tainted" will become 1. If the three conditions have different impact on the mission's taint status, the numbers in the CPT table can be modified accordingly to reflect such difference. For example, in Table 3, "Task 1 is tainted" has greater impact on missions than the other two conditions. When "Task 1 is tainted", the probability for the mission being tainted is bigger than 0.9, no matter if the other conditions are true or false. When Task 1 is not tainted, the probability for the mission being tainted is very low, even if task 2 is tainted or the operation sequence is incorrect. The CPT table can also be modified to accommodate other noise factors that cannot be completely taken into consideration. For example, in Table 3, even if all the three conditions are true, the probability of mission 1 being tainted may not be 1, but a number very close to 1, such as 0.99.

After the BN is constructed, the taint status of system objects is input into BN as evidence. The BN then computes the probabilities of missions being infected based on the given evidence.

5. RELATED WORK

Mission Impact Assessment. Some high level frameworks and models have been established in recent studies to enable qualitative evaluation towards cyber attacks' impact on missions. Alberts et al. [15] proposed a Mission Assurance Analysis Protocol (MAAP) to determine how the current conditions can affect a project. Watters et al. [16] proposed a Risk-to-Mission Assessment Process to map the network nodes to the business objectives. Musman et al. [14] clarified the cyber mission impact assessment framework and related the business processes with technology capacities. Dai et al. [2] proposed a Situation Knowledge Reference Model (SKRM) that enables capabilities such as asset classification, mission damage and impact assessment. [1] is one of the few works that explore quantitative mission impact assessment. It presented an impact-oriented cyber attack model, where an attack has an impact factor and the asset is measured with operational capacity. The assets' operational capacity will be affected by the attack's impact factor. The paper then briefly introduced the impact dependency graph (IDG), but didn't provide details for the construction method.

Bayesian Network. Bayesian networks have been employed in a number of studies for cyber security defense. [17] presented a BN modeling approach which modeled three uncertainty types in the security analysis process. The BN was constructed on top of the logical attack graphs [18, 19]. [20] proposed to construct a cross-layer Bayesian network to infer stealthy bridges existing between the enterprise network islands in cloud. [21] described a mission-impact-based approach to correlate the security alarms collected from different sensors using Bayesian networks. An incident rank tree was built to calculate the rank of each security alert, which combines the incident's impact towards the mission, and the success probability of the activity reported in the alert. Our work also applies Bayesian networks, but targets a different problem.

6. CONCLUSION

This paper proposed a probabilistic approach to evaluate the impacts towards missions caused by attacks. To associate attacks with system assets, a System Object Dependency Graph (SODG) can be built to reflect the influence of attacks towards system objects and capture the intrusion propagation to other objects as well. To further relate the assets with missions, we proposed to buid a mission-task-asset (MTA) map based on the SODG so that the attacks' impact towards system objects can propagate to the related missions. We provided an example Bayesian network that is constructed on top of the MTA to show how our approach can be applied to infer the probabilities of missions being tainted.

Acknowledgement

This work was supported by NIST 70NANB14H218, ARO W911NF-09-1-0525 (MURI), NSF CNS-1422594, and ARO W911NF-13-1-0421 (MURI).

Disclaimer

Table 2: CPT of Mission 1 in the Figure 6

Mission1	Task 1=Tainted				Task 1=Untainted			
	Task 2=Tainted		Task 2=Untainted		Task 2=Tainted		Task 2=Untainted	
	C = True	C = False	C = True	C = False	C = True	C = False	C = True	C = False
Tainted	1	1	1	1	1	1	1	0
Untainted	0	0	0	0	0	0	0	1

Note: C represents the condition "Task 1 is committed before Task 2"

Table 3: Modified CPT of Mission 1 in the Figure 6

Mission1	Task 1=Tainted				Task 1=Untainted			
	Task 2=Tainted		Task 2=Untainted		Task 2=Tainted		Task 2=Untainted	
	C = True	C = False	C = True	C = False	C = True	C = False	C = True	C = False
Tainted	0.99	0.9	0.9	0.9	0.2	0.2	0.2	0.01
Untainted	0.01	0.1	0.1	0.10	0.8	0.8	0.8	0.99

Note: C represents the condition "Task 1 is committed before Task 2"

Institute of Standards and Technology, nor does it imply that the identified products are necessarily the best available for the purpose.

7. REFERENCES

[1] Gabriel Jakobson. Mission Cyber Security Situation Assessment Using Impact Dependency Graphs.

[2] Jun Dai, Xiaoyan Sun, Peng Liu, Nicklaus Giacobe. Gaining Big Picture Awareness through an Interconnected Cross-layer Situation Knowledge Reference Model. 2012 ASE International Conference on Cyber Security, Washington DC, 2012

[3] Tripwire. http://www.tripwire.com/.

[4] Snort. https://www.snort.org/.

[5] Tcpdump. http://www.tcpdump.org/.

[6] S. T. King, and P. M. Chen. Backtracking intrusions. ACM SIGOPS, 2003.

[7] X. Xiong, X. Jia, and P. Liu. Shelf: Preserving business continuity and availability in an intrusion recovery system. ACSAC, 2009.

[8] J. Dai, X. Sun, and P. Liu. Patrol: Revealing zero-day attack paths through network-wide system object dependencies. ESORICS, 2013.

[9] A. Natarajan, P. Ning, Y. Liu, S. Jajodia, and S.E. Hutchinson. NSDMiner: Automated discovery of Network Service Dependencies. In Proceeding of IEEE International Conference on Computer Communications, 2012.

[10] Barry Peddycord III, Peng Ning, and Sushil Jajodia. On the accurate identifi- cation of network service dependencies in distributed systems. In USENIX Association Proceedings of the 26th international conference on Large Installation System Administration: strategies, tools, and techniques, 2012.

[11] Rodrigo Fonseca, George Porter, Randy H. Katz, Scott Shenker, and Ion Stoica. X-trace: A pervasive network tracing framework. In USENIX Association Proceedings of the 4th USENIX conference on Networked systems design and implementation, 2007.

[12] Paul Barham, Richard Black, Moises Goldszmidt, Rebecca Isaacs, John Mac- Cormick, Richard Mortier, and Aleksandr Simma. Constellation: automated discovery of service and host dependencies in networked systems. In TechReport MSR-TR-2008-67, 2008.

[13] Jun Dai. Gaining Big Picture Awareness in Enterprise Cyber Security Defense. Ph.D. dissertation, 2014.

[14] S. Musman, A. Temin, M. Tanner, D. Fox, and B. Pridemore. Evaluating the Impact of Cyber Attacks on Missions. MITRE Technical Paper 09-4577, July 2010.

[15] Alberts C., et al. (2005). Mission Assurance Analysis Protocol (MAAP): Assessing Risk in Complex Environments. CMU/SEI-2005-TN-032. Pittsburgh, PA: Carnegie Mellon University.

[16] Watters J., et al. (2009). The Risk-to-Mission Assessment Process (RiskMAP): A Sensitivity Analysis and an Extension to Treat Confidentiality Issues.

[17] P. Xie, J. H. Li, X. Ou, P. Liu, and R. Levy. Using Bayesian networks for cyber security analysis. DSN, 2010.

[18] X. Ou, W. F. Boyer, and M. A. McQueen. A scalable approach to attack graph generation. ACM CCS, 2006.

[19] X. Ou, S. Govindavajhala, and A. W. Appel. MulVAL: A Logic-based Network Security Analyzer. USENIX security, 2005.

[20] Xiaoyan Sun, Jun Dai, Anoop Singhal, Peng Liu. Inferring the Stealthy Bridges between Enterprise Network Islands in Cloud Using Cross-Layer Bayesian Networks 10th International Conference on Security and Privacy in Communication Networks (SecureComm 2014), Beijing, China

[21] M. Fong, P. Porras, and A. Valdes. A Mission-Impact-Based Approach to INFOSEC Alarm Correlation. Proceedings Recent Advances in Intrusion Detection. Zurich, Switzerland, October 2002.