Towards Actionable Mission Impact Assessment in the Context of Cloud Computing

Xiaoyan Sun, Anoop Singhal, Peng Liu

Towards Actionable Mission Impact Assessment in the Context of Cloud Computing

Xiaoyan Sun[1], Anoop Singhal[2], Peng Liu[3]

[1] California State University, Sacramento, CA 95819, USA
[2] National Institute of Standards and Technology, Gaithersburg, MD 20899, USA
[3] Pennsylvania State University, University Park, PA 16802, USA
xiaoyan.sun@csus.edu, anoop.singhal@nist.gov, pliu@ist.psu.edu

Abstract. Today's cyber-attacks towards enterprise networks often undermine and even fail the mission assurance of victim networks. Mission cyber resilience (or active cyber defense) is critical to prevent or minimize negative consequences towards missions. Without effective mission impact assessment, mission cyber resilience cannot be really achieved. However, there is an overlooked gap between mission impact assessment and cyber resilience due to the non-mission-centric nature of current research. This gap is even widened in the context of cloud computing. The gap essentially accounts for the weakest link between missions and attack-resilient systems, and also explains why the existing impact analysis is not really actionable. This paper initiates efforts to bridge this gap, by developing a novel graphical model that interconnects the mission dependency graphs and cloud-level attack graphs. Our case study shows that the new cloud-applicable model is able to bridge the gap between mission impact assessment and cyber resilience. As a result, it can significantly improve the effectiveness of cyber resilience analysis of mission critical systems.

1 Introduction

Due to the increasing severity of cyber-attacks, mission assurance entails critical demands of active cyber defense and cyber resilience more than ever. Especially in the public cloud where a large number of enterprise networks reside, failure of effective cyber defense would generate huge negative impact towards enterprises' missions. Mission cyber resilience or active cyber defense means capabilities to make prioritized, proactive and resource-constraint-aware recommendations on taking cyber defense actions, including network and host hardening actions, quarantine actions, adaptive MTD (Moving Target Defense) actions, roll-back actions, repair and regeneration actions. Due to the fundamental necessity and importance of situational awareness to decision making, cyber situational awareness plays a critical role in achieving mission cyber resilience. Specifically, mission cyber resilience cannot be really achieved without impact assessment. That is, knowing which mission and how a mission is impacted by an attack is the key for making correct resilience decisions.

However, there is actually a largely overlooked gap between mission impact assessment and cyber resilience, though both mission impact assessment and attack-resilient systems have been extensively researched in the literature:

1) Despite extensive research on attack-resilient survivable systems and networks [1], most if not all existing cyber resilience techniques are unfortunately not mission-centric. The cyber resilience analysis is usually constrained to the level of assets, without consideration in regards to the mission impact. For example, the recommendation of having a backup server would be made by performing asset-level resilience analysis, but the mission-level impact is rarely analyzed. Lack of mission models and mission dependency analysis is a common limitation of existing attack resilience techniques. Without mission dependency analysis, existing cyber resilience techniques cannot quantify the effectiveness of the recommended cyber response actions in terms of mission goals, and hence cannot convincingly justify the superiority of the recommended response actions.

2) From another aspect, despite extensive research on mission impact assessment, mission impact assessment results cannot be automatically used to make mission-centric recommendations on taking cyber response actions. For instance, even if the dependency relationship between a server and a mission is definite, it's still difficult to make any resilience recommendations regarding this server due to the absence of asset-level cyber resilience analysis. This gap is even widened in the context of cloud computing. In public cloud, each enterprise network has its own missions. These missions are usually expected to be independent and isolated from each other. However, multi-step attacks may penetrate the boundaries of individual enterprise networks from the same cloud, and thus impact missions of multiple enterprise networks. That is, attacks that happen in one enterprise network may be able to affect missions of another enterprise network in the same cloud. Therefore, mission impact should be re-assessed in cloud environment.

3) In addition, most mission impact assessment techniques are generally one-dimensional, without explicitly considering the dimension of service dependency. For example, a mission dependency graph usually specifies the dependency relations among a task and its supporting services, but may not include dependencies among services. However, service dependency is also indispensable for accurate mission impact analysis. For example, a web service may depend on the authentication service to verify users, while the authentication service may further depend on the database service to query users' credentials. If the database service is down, missions related to the web service and authentication service may also be impacted, although they do not directly depend on the database service. As a result, the service dependency relations should be explicitly included for accurate mission cyber resilience analysis.

Hence, lack of automation tools in associating missions with attack-resilient systems is a weakest link in achieving cyber resilience. Without such association, existing mission impact analysis results are not really actionable: it's difficult to find out why and how a mission has been impacted. Since bridging this gap may

significantly boost the cyber resilience of mission critical systems, how to bridge this gap is a very important problem.

Therefore, the primary objective of this paper is to take the first steps towards systematically bridging the critical gap between mission impact assessment and cyber resilience in the context of cloud computing. We aim to model the mission impact process and enable the automatic reasoning of this process. To achieve this goal, we identified and addressed the following challenges:

First, it is very challenging to envision a never-seen-before graphical model that can integrate mission dependency graphs and cloud-level attack graphs in such a way that can effectively bridge the gap between existing mission impact analysis results and attack-graph-based active cyber defense. No graphical model has yet been proposed to bridge this gap, though two schools of thoughts have been respectively developed on mission impact analysis and attack-graph-based active cyber defense.

Second, a cloud environment gives rise to new challenges in bridging the gap. Cloud services such as Infrastructure as a Service (IaaS), make attack graphs more complicated and harder to get analyzed. Conventional attack graphs cannot capture the stealthy information flows introduced by certain cloud features. Attackers could leverage the hidden security vulnerabilities caused by inappropriate cloud management to launch zero-day attacks.

The significance of this paper's contributions is two-fold: 1) We have developed a novel graphical model, the mission impact graph model, to systematically bridges the critical gap between impact assessment and cyber resilience. Bridging this gap significantly boosts the cyber resilience of mission critical systems; 2) To the best of our knowledge, this is the first work that investigates the mission impact assessment problem by considering the special features in cloud computing environment; 3) We have extended the attack graph generation tool MulVAL [9] to enable logical reasoning of mission impact assessment and automatic generation of mission impact graphs.

2 Our Approach

In this paper, we aim to bridge the gap between mission impact assessment and cyber resilience.

On the side of mission impact assessment, different types of mission dependency graphs have been developed to associate missions with component tasks and assets. As shown in Figure 1, the status of assets (hosts, virtual machines, etc.) will generate direct impact towards missions through dependency relations. In current literature, such dependency relations among assets, tasks, and missions are usually very loose and not well defined. As a result, the corresponding mission impact assessment is also inaccurate. In addition, without considering the possibility of multi-step attacks caused by combinations of vulnerabilities, the mission impact assessment is usually not sufficiently comprehensive. Missions that seem irrelevant to some compromised assets may also be impacted due to the existence of multi-step attacks. For example, in Figure 1, let's assume mis-

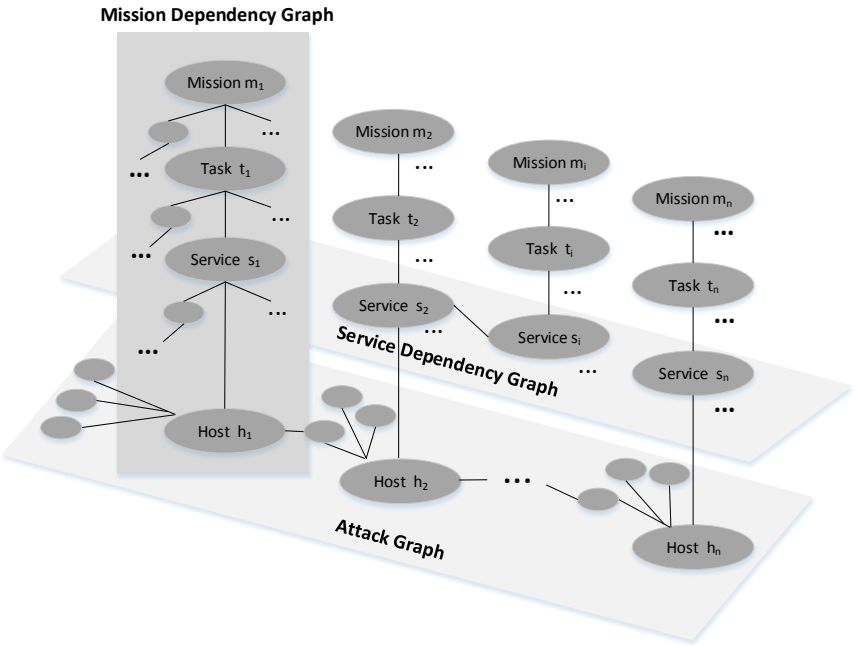

Fig. 1: The Mission Dependency Graph, Service Dependency Graph, and Attack Graph.

sion m_1 and mission m_n depends on host h_1 and host h_n respectively. Analysis of the individual mission dependency graphs (vertical graphs in Figure 1) could lead to a conclusion: if h_1 is compromised, then m_1 will be impacted but m_n will not. This is because the individual mission dependency graphs do not show any direct relations between mission m_n and host h_1. However, if the possibility of a multi-step attack is considered, host h_1 can be used as a stepping-stone to compromise h_n. In this case mission m_n has the possibility of being impacted as well. Therefore, with only mission dependency graphs, it is not sufficient to perform accurate and comprehensive mission impact assessment.

On the side of cyber resilience, attack graphs have become mature techniques for analyzing the causality relationships between vulnerabilities and exploitations. As in Figure 1, by analyzing the vulnerabilities existing in the network, attack graphs are able to generate potential attack paths that show a sequence of attack steps (e.g., from host h_1 to h_n). This capability enables security admins to proactively analyze the influence of some security operations towards the potential attack paths. For example, security admins could check how potential attack paths would be changed if a vulnerability is patched. However, the traditional attack graph has two limitations. First, it is not mission-centric. The attack graph is able to generate potential attack paths through logical reasoning, but it lacks the capability of reasoning potential impacts towards missions. Second, traditional attack graphs do not consider potential attacks enabled by

some special features of public cloud environment, such as virtual machine image sharing and virtual machine co-residency. New attack types may arise in cloud due to these features, but are not captured in traditional attack graphs.

In addition, service dependency discovery has been studied in a number of research works [20–22]. In a network, the normal functioning of an application or service may depend on the normal functioning of other services. Service dependency graphs are able to capture such dependency relations. The service dependencies are important for correct mission impact assessment in that missions may be indirectly impacted by other seem-to-be irrelevant services. For example, mission m_i in Figure 1 does not directly depend on service s_2 in the vertical mission dependency graph. Without considering the service dependencies, m_i seems to be irrelavant with s_2 and thus will not be impacted by its status. However, by taking service dependencies into account, the status of s_2 will affect service s_i, which will eventually impact m_i. Hence, service dependency graphs should also be incorporated when performing mission impact assessment for the purpose of cyber resilience.

Therefore, considering the respective capabilities and disadvantages of mission dependency graphs and attack graphs, this paper proposes to develop a logical graphical model, called *attack graph based mission impact graph* (referred as mission impact graph in the rest of this paper), to integrate mission dependency graphs, service dependency graphs, and cloud-level attack graphs. Our approach contains three steps. First, there exist essential semantic gaps between mission dependency graphs and attack graphs. We identify the semantic gaps and unify the representation of nodes and edges. This makes interconnecting mission dependency graphs and attack graphs feasible. Services are already components of mission dependency graphs, so service dependency graphs are align with mission dependency graphs in terms of semantics. Second, to bridge the gap inside a cloud environment, we extend traditional attack graphs into cloud-level attack graphs. The cloud-level attack graphs are incorporated into new mission impact graphs. Third, we implement a set of interaction rules in MulVAL [8,9] to enable automatic generation of logical mission impact graph.

3 The Semantic Gap Between the Attack Graph And the Mission Dependency Graph

Generally speaking, a mission dependency graph is a mathematical abstraction of assets, services, mission steps (also known as tasks) and missions, and all of their dependencies [6]. A mission dependency graph has five types of nodes, including assets, services, tasks, missions and logical dependency nodes. The logical dependency nodes are basically AND-nodes and OR-nodes that represent logical dependencies among other nodes. The AND-node represents that a parent nodes depends on all of its children nodes. The OR-node denotes that a parent node depends on at least one of its children nodes. For example, a successful task may depend on all of the supporting services being functional, while a complete

mission could require only one of its tasks being fulfilled. Edges in a mission dependency graph represent the interdependencies existing among nodes.

As for the attack graph, it usually shows the potential attack steps leading to an attack goal. Several different types of attack graphs have been developed, such as state enumeration attack graphs [10–12] and dependency attack graphs [13–15]. This paper uses the dependency attack graph for analysis. Figure 2 is part of a simplified attack graph. A traditional attack graph generated by MulVAL is composed of two types of nodes, fact nodes (including primitive fact nodes and derived fact nodes) and derivation nodes (also known as rule nodes). Primitive fact nodes (denoted with rectangles in Figure 2) present objective conditions of the network, such as the network, host, and vulnerability information. Derived fact nodes (denoted with diamonds) are the facts inferred by applying the derivation rule. Each derivation node (denoted with ellipse) represents the application of a derivation rule. The derivation rules are implemented as interaction rules in MulVAL. Simply put, one or more fact nodes could be the preconditions of a derivation node, while the derived fact node is the post-condition of the derivation node. For example, in Figure 2, if node 4 "the attacker has access to the server", node 5 "the server provides a service with an application" and node 6 "the application has a vulnerability" are all satisfied, then the rule in node 7 will take effect and make node 8 become true. That is, attacker is able to execute arbitrary code on the server. In this example, node 4, 5 and 6 are the fact nodes; node 7 is the derivation node; node 8 is the derived fact node.

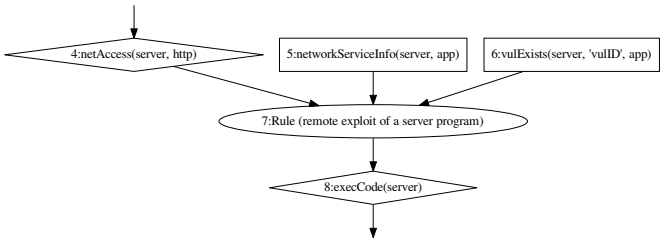

Fig. 2: Part of a Simplified Attack Graph.

Mission dependency graphs and traditional attacks graphs have the following semantic gaps:

1) The meaning of nodes differs. In a mission dependency graph, a node denotes an entity, such as an asset, a service, a task, or a mission. The node does not specify the status of the entity. In a traditional attack graph, a node represents a statement, be it a rule or a fact. For example, a primitive fact node could be "the web server provides OpenSSL service" or "the openssl program has a vulnerability called CVE-2008-0166". A rule node could be "the remote exploit of a server program could happen".

2) The meaning of edges differs. In a mission dependency graph, the edges represent general interdependencies among nodes, and do not specify concrete dependency types. The logical relations are specially denoted with AND and OR nodes. In a traditional attack graph, directed edges represent the causality relationship among nodes. One or more fact nodes could cause a derivation node to take effect, which further enables a derived fact node.

3) The representation of logical relations among nodes differs. In a mission dependency graph, the logical relations are represented specifically with AND and OR nodes. In traditional attack graph, the logical relations are not provided explicitly, but are implied in the graph structure: derivation nodes (rule nodes) imply AND relations and derived fact nodes imply OR relations. That is, fact nodes that serve as preconditions of a derivation node have AND relations, while derivation nodes leading to a derived fact node have OR relations. The underlying principle is that all of the preconditions have to be satisfied to enable a derivation rule, while a derived fact node can become true as long as one rule is satisfied.

Hence, in the proposed mission impact graph, we will bridge the semantic gaps between mission dependency graphs and attack graphs.

4 Incorporating Cloud-level Attack Graphs

In the public cloud, each enterprise network can generate its own individual attack graph by scanning hosts and virtual machines in the network. These individual graphs may not be complete because new attack paths enabled by the cloud environment could be missed. Therefore, a cloud-level attack graph is needed to capture potential missing attacks by taking some features of public cloud into consideration, such as virtual machine image sharing and virtual machine co-residency. Hence, [16] proposed the construction of cloud-level attack graphs. A cloud-level attack graph contains three levels: virtual machine level, virtual machine image level, and host level. The virtual machine level mainly captures the causality relationship between vulnerabilities and potential exploits inside the virtual machines. The virtual machine image level focuses on attacks related to virtual machine images. For example, a virtual machine image may be instantiated by different enterprise networks. As a result, its security holes are also inherited by all the instance virtual machines. The virtual machine image level is able to reflect such inheritance relationship. The host level mainly captures potential attacks to hosts, including exploits leveraging the virtual machine co-residency relationship.

Therefore, the mission impact graph needs to be extended to incorporate cloud-level attack graphs. The semantics of mission impact graphs remain the same because cloud-level attack graphs have the same semantics as traditional attack graphs. However, the mission impact graph is now composed of two parts: cloud-level attack graph part, and the cloud-applicable mission dependency part. New nodes should be added as derivation nodes and fact nodes to incorporate special features of cloud.

To achieve this goal, we crafted a set of Datalog clauses in MulVAL as the primitive facts, derived facts and interaction rules. For the cloud-level attack graph part, new facts and rules are crafted to model virtual machine image vulnerability existence, vulnerability inheritance, backdoor problem, and virtual machine co-residency problem, and so on. For mission dependency part, new rules are added to model the residency dependencies among virtual machines and hosts, service dependencies among virtual machines and services, etc. For example, the residency dependency relationship between a host and the dependent virtual machines can be modeled with the following interaction rule:

```
interaction rule(
   (hostImpact(VM):-
      residencyDepend(Vm, Host),
      HostImpact(Host)),
   rule_desc('An compromised host will impact the dependent
virtual machines')).
```

5 Mission Impact Graph and Graph Generation

The new graphical model, which is referred to as mission impact graph, is formally defined as follows: 1) It is a directed graph that is composed of three parts: attack graph part, service dependency part and mission-task-service-host dependency part. 2) It contains two kinds of nodes: derivation nodes and fact nodes. Each fact node represents a logical statement. Each derivation node represents an interaction rule that is applied for derivation. There are two types of fact nodes, primitive fact nodes and derived fact nodes. A primitive fact node represents a piece of given information, such as host configuration, vulnerability information, network connectivity, service information, progress status of a mission (e.g. which mission steps are already completed and which are not), and so on. Derived fact nodes are computing results of applying interaction rules iteratively on input facts. 3) The edges in the mission impact graph represent the causality relations among nodes. A derived fact node depends on one or more derivation nodes (which have OR relations); a derivation node depends on one or more fact nodes (which have AND relations).

In mission impact graphs, we need to combine attack graphs and mission dependency graphs by unifying their representation of nodes and edges. It is composed of four steps:

Step 1, the entity nodes in mission dependency graphs become either primitive fact nodes or derived fact nodes in mission impact graphs. A fact node represent a statement about an entity. Primitive fact nodes usually represent already known information provided by network scanners or human administrators, such as host configuration, network configuration, and vulnerability information, etc. Derived fact nodes are computed information by applying interaction rules towards the primitive fact nodes. One entity in the mission dependency graph may become a number of fact nodes depending on its related known information and computed information. For example, a service node s in the mission dependency graph may be related to two pieces of information: the known information that

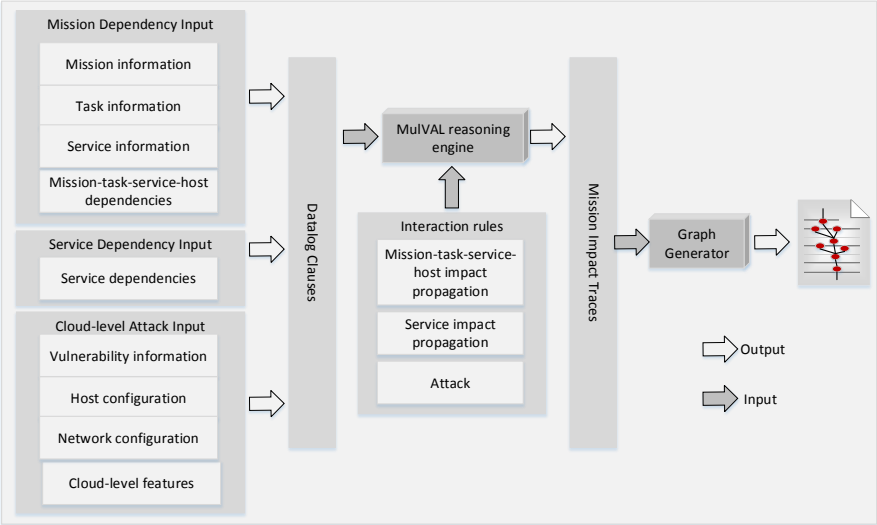

Fig. 3: Logical Mission Impact Graph Generation.

showing "s is disabled" through a port scanning, and the computed information that "s is impacted" by applying interaction rules. In this case, the s node in the mission dependency graph could become two fact nodes in the mission impact graph: a primitive fact node with the predicate *serviceDisabled*; and a derived fact node with the predicate *serviceImpacted*. Similarly, an asset node h may in mission dependency graph could become a derived fact node with predicate *execCode*, meaning "attackers can execute arbitrary code on the host h".

Step 2, derivation nodes are added into the mission impact graph to model the causality relationships among fact nodes. The interdependencies among entities such as assets, services, tasks, and missions in the mission dependency graph can be interpreted into specific impact causality rules, which become derivation nodes in mission impact graph. For example, the dependency between a task t and a service s could be interpreted into a rule R: "t will be compromised if s is disabled and t is not completed yet". When node "s is disabled" and node "t is not completed yet" are both satisfied, the derivation node stating rule R will take effect.

Step 3, logical relation nodes in mission dependency graphs are removed, including AND and OR nodes. The logical relations among nodes are implied with graph structure as in the mission impact graph: derivation nodes imply AND, and derived fact nodes imply OR.

Step 4, the fact nodes and derivation nodes in mission impact graphs are connected with edges to represent direct causality relations, rather than general dependencies as in mission dependency graphs.

Finally, to enable automatic generation of mission impact graphs, we extended the capability of MulVAL by creating new Datalog clauses. Figure 3 shows the process of mission impact graph generation. Three sets of input are provided to MulVAL in terms of mission dependencies, service dependencies, and cloud-level attacks. The input sets are converted to corresponding Datalog clauses, which are then fed to MulVAL reasoning engine. The MulVAL reasoning engine analyzes the input sets and perform computation by applying interaction rules. Interaction rules are very important for the logical reasoning. For mission impact analysis, we created three different sets of interaction rules, including rules for mission-task-service-host impact propagation, service impact propagation, and attack analysis. After reasoning, mission impact traces are generated. Graph generators such as Graphviz [23] can analyze the traces to build the final mission impact graphs. Originally MulVAL only contains the input set and interation rules for attack graph generation. We added two more input sets and two more interation rule sets, and also extended the attack-related input set and interaction rule set by including cloud-level features.

In the implementation, three sets of Datalog clauses are added as primitive facts, derived facts, and interaction rules for the function of mission impact analysis. For primitive facts, we crafted clauses that describe mission-task dependencies, the service types, task service dependencies, and mission progress status, and so on. Some information can be provided by system administrators. For derived facts, we added clauses for the status of missions, tasks, services and assets. To enable the logical reasoning, we created interaction rules to model the causality relationships between pre-conditions and post-conditions. For example, attacks towards servers will impact services that are provided by these servers. The interaction rule describing this causality relationship could be represented with Datalog clauses as follows:

```
interaction rule(
   (serviceImpacted(Service, H, Perm):-
      hostProvideService(H, Service),
      execCode(H, Perm)),
   rule_desc('An compromised server will impact the dependent
service')).
```

6 Case Study

As shown in Figure 4, our scenario contains three enterprise networks in cloud: A is a start-up company, B is a medical group, and C is a vaccine supplier. In addition to providing existing vaccines, C is also developing a new type of vaccine together with its collaborators. The formula of the new vaccine is still very confidential. For security purposes, C only accepts client requests from trusted IPs. The relationships between A, B and C are: 1) they are on the same cloud; 2) A's webserver and B's database server are two virtual machines that co-reside on the same physical host; This is not uncommon in that cloud providers generally host virtual machines on arbitrary hosts. This co-residency relationship can be leveraged by attackers. 3) B is a trusted client to C.

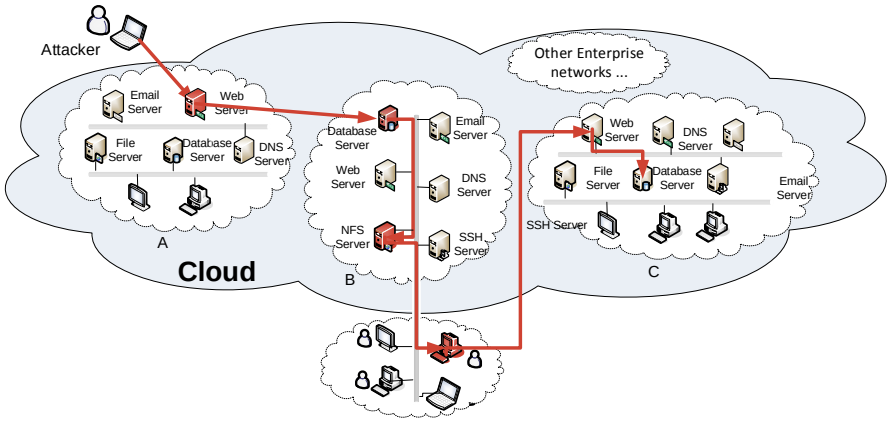

Fig. 4: The Attack and Mission Scenario.

The mission for medical group B, *Bm1*, is to provide medical services to all of its patients. Sample tasks include: *Bt1*, patients make appointments; *Bt2*, access medical records; *Bt3*, order shots or medicine; *Bt4*, administer shots; *Bt5*, update medical records, and so on. The mission for vaccine supplier C, *Cm1*, is to supply vaccines to authorized medical groups, and develop the new type of vaccine with collaborators. The sample tasks include: *Ct1*, ask for login ID and password; *Ct2*, check the ID and password. If the user is medical group, go to *Ct3*. If the user is collaboration partner, go to *Ct4*; *Ct3*, order vaccine; *Ct4*, check and update new vaccine information. *Ct3* and *Ct4* are then composed of a number of subtasks.

In our attack scenario, attacker Mallory (could be a competitor of victim companies) is very interested in the new vaccine, and wants to steal its formula from supplier C. To break into the supplier network, Mallory performs the following attack steps: 1) Mallory compromises A's webserver by exploiting vulnerability CVE-2007-5423 in tikiwiki 1.9.8; 2) Mallory leverages the co-residency relationship to take over B's database server, based on a side channel attack in cloud. Extensive research has been done on side channel attacks [17–19]. 3) B's NFS server has a directory (*/exports*) that is shared by all the servers and workstations inside the company. Normally B's web server should not have write permission to this shared directory. However, due to a configuration error of the NFS export table, the web server is given write permission. Therefore, Mallory uploads a Trojan horse to the shared directory, which is crafted as a management software named *tool.deb*. 4) The innocent Workstation user from B downloads *tool.deb* from NFS server and installs it. This creates an unsolicited connection back to Mallory. 5) The Workstation has access to C's webserver as a trusted client. Mallory then managed to take over it via a brute-force key guessing attack (CVE-2008-0166); 6) Mallory leverages C's webserver as a stepping stone to compromise C's MongoDB database server based on CVE-2013-1892, which allows Mallory successfully steal credential information from an employee lo-

gin database table; 7) Mallory logins into C's webserver as a collaborator of C, and accesses the project proprietary documentation to collect formula-related vaccine research and development records.

Fig. 5: Mission Impact Graph.

By performing logical reasoning in MulVAL, we generated a mission impact graph for our scenario. Figure 5 shows a part of the graph. MulVAL takes several types of inputs, including vulnerability-scanning report, host configuration, network connectivity, mission-task-service-asset dependencies, and so on. The output is a mission impact graph showing which missions are likely to be affected by considering current status of the networks.

The result cloud-level mission impact graph is very helpful for understanding potential threats to missions in this scenario. First, individual mission impact graph may miss important attacks leveraging some features of cloud, and thus generate incorrect evaluation about possible threats to missions. For example, without considering the co-residency relationship between A's webserver and B's database server, B seems to be very safe as the database has no exploitable vulnerability. As a result, mission *Bm1* is viewed as safe. However, our mission impact graph shows that *Bm1* has the possibility of being impacted because the virtual machine co-residency can be leveraged for attack. Second, the result cloud-level mission impact graph is mission-centric. Although attack paths for the scenario network can be generated in the traditional attack graph, the potential impact towards mission cannot be assessed without analyzing the de-

pendency relationships among missions, tasks, services and assets. For example, the attack graph is able to show the attack path from A's web server to C's database server, but the impact of attacks to missions is unclear. Our mission impact graph is able to show such impact towards missions by considering both attacks and missions.

One function of our mission impact graph is to perform automated "taint" propagation through logical reasoning. Given a "taint", be it a vulnerability, a compromised machine, or a disabled service, the impact of the "taint" can be analyzed through logical reasoning. The mission impact graph is able to reflect affected entities such as assets, services, tasks, and missions. For example, in our case study, if C's web server is compromised, the mission impact graph will show that task *Ct1* and mission *Cm1* are impacted.

The generated mission impact graph enables effective mission-centric cyber resilience analysis. Propagating the attack-graph-based active cyber defense from the attack graph side to the mission impact side is helpful for performing advanced proactive "what-if" mission impact assessment. Through logical reasoning, impact analysis can be performed all the way from inside a machine to a mission. Given the input information, attack graphs can predict the potential attack paths and identify possibly to-be-affected assets. Mission impact graph extends attack graphs in a way that the prediction of potential attack paths directly enables the prediction of potential mission impact. Therefore, we can perform proactive "what-if" mission impact analysis by changing the input conditions. For example, what if we remove a server? What if we patch a vulnerability on a host? Which tasks or missions will be affected? In our case study, if we break the co-residency relationship between A's webserver and B's database server by moving one of the virtual machines, attacks towards B and C will prevented. As a result, missions in B and C won't be affected. Similarly, if the vulnerability on C's webserver is patched, attacks towards C can be stopped and mission *Cm2* will be safe. In addition, we can also analyze the potential mission impact by assuming vulnerability existence on other servers. For example, what if an unknown security hole exists on a host? Which tasks or missions will be affected in this case? In addition, as the situation knowledge regarding a network is continuously collected, such knowledge can be interpreted into input files to the automated tool for iterative "what-if" mission impact analysis based on the current situation. Therefore, performing such "what-if" analysis enables interactive mission impact analysis, and thus helps security admins make correct decisions for cyber resilience.

7 Related Work

A literature review is performed to disclose the mismatch between mission impact assessment and cyber resilience: 1) the formal models used by the existing mission impact assessment techniques cannot be directly used by the existing attack-resilient system and network designs; 2) lack of mission models and mis-

sion dependency analysis is a common limitation of existing cyber resilience techniques.

Mission impact assessment. In the past decade or so, extensive research has been conducted on modeling the mission dependencies to help facilitate computer-assisted analysis of current missions. The existing mission-oriented impact assessment techniques can be classified into four categories: 1) mission impact assessment through use of ontology based data collection. The basic idea is to create the ontology of mission dependencies. For example, the Cyber Assets to Mission and Users (CAMUs) approach [2] assumes that a cyber asset provides a cyber capability that in turn supports a mission. Their approaches mine existing logs and configurations, such as those from LDAP, NetFlow, FTP, and UNIX to create these mission-asset mappings; 2) mission impact assessment through use of dependency graphs [4, 5]. The basic idea is the use of mission dependency graphs for cyber impact assessment and a hierarchical (time-based) approach to mission modeling and assessment; 3) mission impact assessment through use of mission thread modeling [6]. The basic idea is to leverage mission metrics supported by resource model and value model; 4) mission impact assessment through use of Yager's aggregators [3]. The basic idea is to utilize a tree-based approach to calculate the impact of missions. The mission tree is a tree-structure that utilizes Yager's aggregators [7] to intelligently aggregate the damage of assets to calculate the impact on each individual mission.

Cyber resilience and active cyber defense. Since 2000, a tremendous amount of research has been conducted on how to make systems and networks resilient to cyber-attacks. For example, the two volumes of DARPA Information Survivability Conference and Exposition proceedings described the design, implementation and evaluation of the first set of survivable and attack-resilient systems and networks [1]. The cyber response actions adopted in these systems include replication actions, honeypot actions, software diversification actions, dynamic quarantine actions, adaptive defense actions, roll-back actions, proactive and reactive recovery actions. Since then, a variety of cyber response actions have emerged, including migration actions, regeneration actions, MTD actions, decoy actions, CFI (control flow integrity) actions, ASLR (Address Space Layout Randomization) actions, IP randomization actions, N-variant defense actions, and software-defined network virtualization actions.

8 Conclusion

This paper makes the first efforts to close a gap between mission impact assessment and cyber resilience. In the cloud environment it is even more difficult to analyze the impact of vulnerabilities and security events on missions. To fill the gap and associate missions with current attack-resilient systems, this paper develops a novel graphical model that interconnects mission dependency graphs and cloud-level attack graphs. Our case study shows that the mission impact graph model successfully bridges the gap and can significantly boost the cyber resilience analysis of mission critical systems.

Acknowledgement

We thank the anonymous reviewers for their valuable comments. This work was supported by ARO W911NF-15-1-0576, ARO W911NF-13-1-0421 (MURI), CNS-1422594, NIETP CAE Cybersecurity Grant, and NIST 60NANB16D241.

Disclaimer

References

1. Proceedings of DARPA Information Survivability Conference and Exposition, Anaheim, California, 12-14 June 2001, Volume I & Volume II.
2. A. D'Amico, L. Buchanan, J. Goodall. Mission Impact of Cyber Events: Scenarios and Ontology to Express the Relationships between Cyber Assets, Missions, and Users. Proc. 5th Int'l Conf. on Information Warfare and Security, 2010.
3. J. Holsopple, S. J. Yang, and M. Sudit. Mission Impact Assessment for Cyber Warfare. Book chapter in Intelligent Methods for Cyber Warfare, Springer, 2015.
4. Gabriel Jakobson. Mission Cyber Security Situation Assessment Using Impact Dependency Graphs. In Information Fusion (FUSION), 2011.
5. R. Sawilla, X. Ou. Identifying critical attack assets in dependency attack graphs. Defense R&D Canada-Ottawa, Technical Memorandum, DRDC Ottawa TM 2008-180, 2008.
6. S. Musman, A. Temin, M. Tanner, D. Fox, B. Pridemore. Evaluating the Impact of Cyber Attacks on Missions. MITRE Corporation, 2009.
7. R. R. Yager. On ordered weighted averaging aggregation operation in multicriteria decision making. IEEE Transactions on Systems, Man and Cybernetics, Vol. 18, pages 183-190, 1988.
8. X. Ou, W. F. Boyer, and M. A. McQueen. A scalable approach to attack graph generation. ACM CCS, 2006.
9. X. Ou, S. Govindavajhala, and A. W. Appel. MulVAL: A Logic-based Network Security Analyzer. USENIX security, 2005.
10. O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. Automated generation and analysis of attack graphs, in Security and Privacy (S&P), 2002.
11. C. R. Ramakrishnan, R. Sekar, Model-based analysis of configuration vulnerabilities, Journal of Computer Security, 2002.
12. C. Phillips and L. P. Swiler, A graph-based system for network-vulnerability analysis, in Proceedings of workshop on New security paradigms, 1998.
13. S. Jajodia, S. Noel, and B. O'Berry, Topological analysis of network attack vulnerability, Managing Cyber Threats, 2005.
14. P. Ammann, D. Wijesekera, and S. Kaushik. Scalable graph-based network vulnerability analysis. ACM CCS, 2002.

15. K. Ingols, R. Lippmann, and K. Piwowarski. Practical attack graph generation for network defense. ACSAC, 2006.
16. Xiaoyan Sun, Jun Dai, Anoop Singhal, Peng Liu. Inferring the Stealthy Bridges between Enterprise Network Islands in Cloud Using Cross-Layer Bayesian Networks. SecureComm, 2014.
17. Zhang, Yinqian, et al. Homealone: Co-residency detection in the cloud via side-channel analysis. 2011 IEEE Symposium on Security and Privacy. IEEE, 2011.
18. Ristenpart, Thomas, et al. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. Proceedings of the 16th ACM conference on Computer and communications security. ACM, 2009.
19. Younis, Younis, Kashif Kifayat, and Madjid Merabti. Cache side-channel attacks in cloud computing. International Conference on Cloud Security Management (ICCSM). 2014.
20. Chen, Xu, Ming Zhang, Zhuoqing Morley Mao, and Paramvir Bahl. Automating Network Application Dependency Discovery: Experiences, Limitations, and New Solutions. In Proceedings of the 8th USENIX conference on Operating systems design and implementation (OSDI), 2008.
21. 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.
22. Barry Peddycord III, Peng Ning, and Sushil Jajodia. On the accurate identification 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.
23. http://www.graphviz.org/.