Gaining Big Picture Awareness through an Interconnected Cross-layer Situation Knowledge Reference Model

Jun Dai, Xiaoyan Sun, Peng Liu, Nicklaus Giacobe

College of Information Sciences and Technology
Pennsylvania State University
University Park, PA 16802, USA
{jqd5187, xzs5052, pliu, nxg13}@ist.psu.edu

***Abstract*—In both military operations and the commercial world, cyber situation awareness (SA) is a key element of mission assurance. Due to the needs for mission damage and impact assessment and asset identification (and prioritization), cyber SA is beyond intrusion detection and attack graph analysis. In this paper, we propose a cross-layer situation knowledge reference model (SKRM) to address the unique cyber SA needs of real-world missions. SKRM provides new insight on how to break the "stovepipes" created by isolated situation knowledge collectors and gain comprehensive level big picture awareness. Through a concrete case study, we show that SKRM is the key enabler for two SA capabilities beyond intrusion detection and attack graph analysis. The potentials and the current limitations of SKRM and SKRM-enabled analysis are also discussed.**

Keywords—Cyber situation awareness, mission-driven analytics, damage and impact assessment, asset identification and prioritization

I. INTRODUCTION

A. Cyber Situation Awareness

Organizations rely on business intelligence (BI) software to eliminate the "data rich but information poor" problem and gain organizational Situation Awareness (SA) [16]. Similarly, SA is a necessary pre-requisite for military or enterprise organizations to achieve "intelligence" for cyber security analysis [12]. To better secure a network, human decision makers should clearly know what is going on in the network. This is basically what we call cyber situation awareness (cyber SA). Like traditional SA, cyber SA also contains four levels: perception, comprehension, projection [8] and resolution [17]. McGuinness and Foy [17] use an analogy to explain these four levels: perception represents "What are the current facts?" Comprehension means, "What is actually going on?" Projection asks, "What is most likely to happen if …?" And Resolution means, "What exactly shall I do?"

Achieving complete and accurate cyber SA is critical for security analysis. Technologies regarding cyber security have made remarkable progress in the past decades. A lot of algorithms and tools are developed for vulnerability analysis, intrusion detection, damage and impact assessment, and system recovery, etc. However, much of the work in security analysis is from isolated *microscopic* perspectives, but do not explicitly consider how to enhance human security analysts' overall cyber situation awareness. Tons of data and information are presented to security analysts from all different levels such as business, network, or system, but how can the information be combined to extract the most critical knowledge of current security situation for human analysts' easy digestion and understanding? There are experts from different areas for the same security topic, but how can they effectively communicate with each other? For example, experienced business managers can immediately notice unusual financial loss, but they definitely cannot tell whether the loss is caused by a buffer overflow attack towards one workstation. In the opposite direction, an operating system expert can quickly notice an abnormal system call from the log, but cannot tell how this could affect the business flow of the company. Therefore, a *macroscopic* framework should be established to integrate cyber knowledge from different perspectives by coupling data, information, algorithms and tools, and human knowledge, to enhance cyber analysts' situation awareness.

The four levels of cyber SA is expected to enable capabilities such as mission damage and impact assessment, and asset identification (and prioritization) for real-world mission assurance. Mission damage and impact assessment is essential to identify and track the relevant causality relationships during attacks and subsequent damage propagation. Besides, mission asset analysis is expected to identify and classify critical mission assets into categories like "polluted", "clean but in danger", and "clean and safe". These capabilities will greatly ease the decision making process and facilitate network security management. However, such cyber SA capabilities should be based on big-picture awareness and are hindered by the following "stovepipe" problem.

B. The "Stovepipe" Problem

Both of the above two capabilities require perception, comprehension, projection and resolution of the involved (data and code) elements at different abstraction levels of the computer and information system semantics. Specifically, damage can be identified at the business process level, application/service level, operating system object (file or process) level or instruction level (memory unit, instruction, register and disk sector). However, for a more comprehensive assessment, the capability needs to be based on a multi-level understanding and cross-level awareness. For example, system experts exactly know which file is stolen or modified, but they hardly know how this can impact the business level. On the other hand, business managers can rapidly notice a suspicious financial loss, but they won't relate it to an un-allowed system call parameter inside the operating system. That is, current security solutions are usually restricted and isolated by their corresponding abstraction levels, such as workflow healing [30], intrusion detection [3, 22], attack graph analysis [11, 21, 25, 28], OS-level dependency tracking [15] and recovery [29], and instruction-level taint analysis [13, 19, 31]. When they come to cyber SA, we need them to effectively "talk" to each other and help security analysts achieve overall situation awareness.

In this paper, we refer to such "*isolation*" between different knowledge bases as "*stovepipe" problem*. The above-mentioned abstraction levels are one kind of "stovepipes", because they cause security analytics bound to individual levels of semantics abstraction. Desired capabilities are also hindered by other stovepipes scattered in existing security analytics. For example, this problem is compounded by non-integrated security tools such as vulnerability scanners, anti-virus/malware sensors, monitors

and loggers, IDS, and compartmentalization (processes, physical machines, network segments) on the same abstraction level. Actually, compared with the abstraction levels as *horizontal stovepipes*, this second kind of "stovepipes" are *vertical stovepipes*. For example, hosts in an enterprise network could function as isolated vertical stovepipes, causing some security solutions to be "stovepiped". Specifically, although workflow management and attack graph analysis are inherently able to cover the whole network, technologies such as OS-level dependency tracking and instruction-level taint analysis were originally designed to only cover a single host.

As a result, today's security diagnosis and analytics, such as those based on intrusion detection and attack graph analysis, are inherently stovepiped, either horizontally or vertically. SAs extracted from these individual situation knowledge collectors are just isolated "pieces". Desired large picture outcomes cannot be gained without a holistic understanding of the entire scenario. Therefore these individual elements must be stitched together in an interconnected cross-layer "big picture", which we call "*big picture awareness*". This broader view of cyber SA goes beyond intrusion detection and attack graph analysis and becomes a key enabler of capabilities essential for future cyber operations.

C. Major Challenges

To enable the capability of mission damage and impact assessment and asset identification (and prioritization), we have to gain big picture awareness. To gain big picture awareness, we have to first address the stovepipe problem. Several non-trivial challenges exist in accessing and modeling cross-layer data and applying it in mission-driven analytics:

- First, we have to identify the stovepipes existing in an enterprise network, including horizontal (e.g., abstraction levels) and vertical (e.g., compartments).
- Second, we have to recognize or develop inter-compartment and cross-abstraction-level analytical technologies to break the corresponding stovepipes.
- Third, we have to integrate the isolated network situation knowledge into a "big picture" based on the above two breakthroughs, and further enable the "big-picture oriented" diagnosis.

D. Approach and Contributions

In a word, *the availability of sea of sensed information opens up fascinating opportunities to understand both mission and adversary activity through modeling and analytics.* This will require creative mission-aware analysis of heterogeneous data with cross-compartment and cross-abstraction-layer dependencies in the presence of significant uncertainty and untrustworthiness. This paper is an effort for cross-compartment and cross-abstraction-level integration to gain big picture awareness. To address the above-mentioned challenges, we present an enterprise network situation knowledge reference model (SKRM, shown in Fig. 1) with multiple abstraction layers. Basically, *SKRM is a framework that integrates cyber knowledge from different perspectives by coupling data, information, algorithms and tools, and human knowledge, to enhance cyber analysts' situation awareness*. SKRM is a natural yet critical "next-step" we achieve towards gaining big picture awareness. Our main contributions can be presented as follows:

1) We categorize enterprise network situation knowledge and thus identify the abstraction layers of SKRM: the Workflow Layer, App/Service Layer, Operating System Layer and Instruction Layer. Each layer, from top to bottom, is characterized by additional fine-grained granularity. We regard these abstraction layers as horizontal stovepipes and the compartments (business tasks, applications or services, processes or files, memory units or disk sectors, etc.) on the corresponding abstraction layer as vertical stovepipes.

2) To break these stovepipes, we introduce, extend or develop diagnosis technologies into the mission-driven analytics. For example, to leverage the information in the compartments, inter-compartment data or control dependency tracking technologies are brought and extended into SKRM. These include workflow data or control dependency mining [1, 18, 30], service dependency discovery [5, 6], OS-level dependency tracking [15, 29], and instruction-level taint tracking [13, 19, 31].

   To break the horizontal stovepipes (e.g., abstraction layers), cross-abstraction-layer semantics bridging technologies are introduced or developed into SKRM. Basically, these capture cross-layer (mapping, translation or causality) relationships in-between different levels of computer and information system semantics. For instance, as shown in Fig. 6, a logical dependency Attack Graph [21, 24, 28] is vertically inserted between the App/Service Layer and Operating System Layer. This attack graph enables causality representation and tracking between network service level pre-conditions (configuration and vulnerability information) and identification of successful exploits at the OS level.

3) Based on the above two breakthroughs, we further propose and formalize the multi-layer enterprise network SKRM model as the integrator of isolated network situation knowledge. As such, SKRM proves its value as an enabler of cross-layer diagnosis capability like mission damage and impact assessment and asset identification (and prioritization). As a new type of SA support system, SKRM breaks both vertical (e.g., compartments) and horizontal stovepipes (e.g., abstraction levels), through inter-compartment and cross-abstraction-layer interconnection, respectively. SKRM enables both inter-compartment analysis and cross-layer diagnosis, and thus support big picture awareness. Each abstraction layer of SKRM covers the entire network to integrate inter-compartment awareness of all the mission assets at that layer. Each layer generates a graph that provides us a view of the same network from a different perspective and granularity. SKRM also captures the cross-layer relationships (e.g., mapping, translation, and semantics bridging) to offer cross-layer diagnosis, which will be the "soul" of this model. SKRM results in a graph stack composed of all the graphs and the cross-layer edges, in an integrated fashion, transforming isolated SA into a shared, scalable, stovepipe-breaking and "big-picture-oriented" SA.

In this paper, we propose and formalize the SKRM model and perform a concrete case study based on the business/mission scenario shown in Fig. 7 under a 3-step attack. We deploy various sensors or detectors to collect real data and explain how to generate the SKRM graph stack, as well as how to use it to perform systematic cross-layer analysis. The analytical results show that SKRM is indeed the key enabler of mission damage and impact assessment and asset identification (and prioritization) to support SA, beyond intrusion detection and attack graph analysis. In our conclusion, the potentials and current limitations of SKRM and SKRM-enabled diagnosis are further discussed.

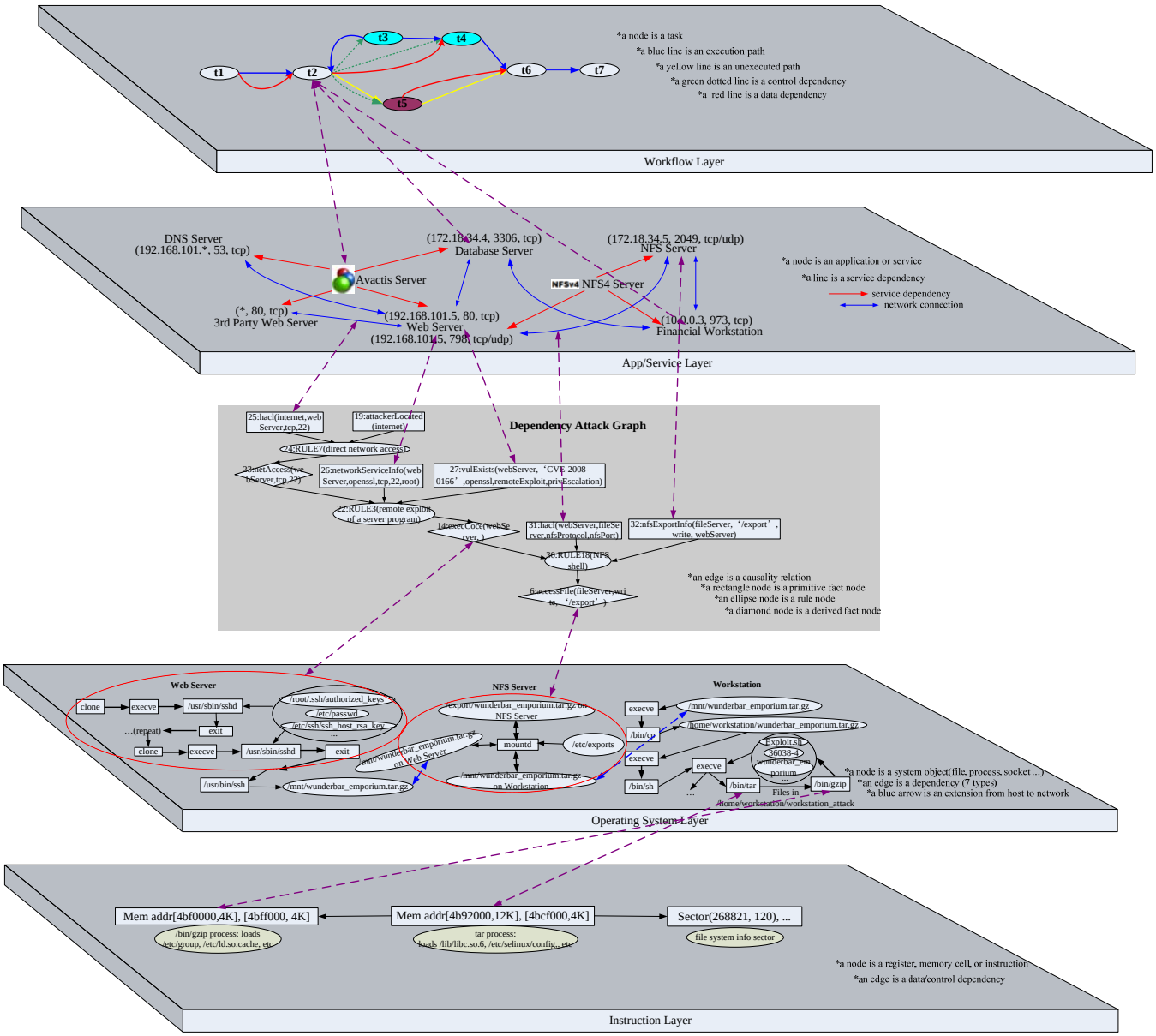

Fig. 1 The proposed enterprise network Situation Knowledge Reference Model (SKRM)

II. Related Work

Different security technologies and tools inspect and handle different aspects of security problems, and each one has something in which it excels. For example, the workflow management technology investigates the data and control dependencies of tasks at the business process level and enables intrusion recovery; network or vulnerability scanners such as Nmap, OVAL and Nessus are powerful to explore useful information about operating systems, ports, connections and exploits; network packet interception and traffic monitoring technologies with Wireshark [27] and Ntop [20] are forceful to look into the network status; IDS systems like Bro [22] and Snort [26] provide alerts when network traffic matches known patterns; attack graph analysis like MulVAL reveals causality relationships between configurations, vulnerabilities and exploits and thus can be used to locate the attack paths; OS-level dependency tracking classifies OS-level objects into files and processes, and tracks backward or forward dependencies between them to identify intrusion roots; instruction-level taint analysis classifies instruction level entities into memory cells, registers, disk sectors, etc., and captures the fine-grained data or control dependencies between them caused by instruction flows [31]. However, these security technologies are isolated because they are not designed to automatically connect to each other by nature. The diagnosis based on these individual elements provides only perception (Level 1 SA [8]). Desired outcomes and capabilities such as mission damage and impact assessment and asset identification (and prioritization) are not adequately supported without a comprehension of the larger picture (Level 2 SA [8]).

Therefore, the above-mentioned stovepipes exist between the current security technologies and hinder the capabilities for mission assurance. Some security products and research in the community have made progress from isolated technologies to

holistic picture such as ArcSight [2], QualysGuard [23], PEDA[31] and the research for managing business health under malicious attacks[32], but the stovepipe problem has not been completely resolved. Specifically, as a leading enterprise security information and event management (SIEM) system, ArcSight provides a correlation engine to visualize the security management of user activities, event logs and intrusion alerts. As a web-based vulnerability scanner, QualysGuard delivers IT security and compliance as a service to combine the business-level view and the network-level view. The work in PEDA generates an Instruction Layer and performs cross-layer infection diagnosis to bridge the "semantic gap" between the Instruction Layer and Operating System Layer [31]. Also, a framework is proposed in [32] to enable the impact analysis of the business-level implications of attacks on the underlying IT systems in real enterprises. All these outcomes are only partial solutions to gain big picture awareness.

Today's existing security solutions are still inherently limited by data isolation, which has not been adequately addressed by current technologies. This paper addresses this issue via breaking such isolation and integrating different perspectives and granularities seamlessly into the SKRM model.

III. SKRM Model

A. SKRM Overview and Features

As shown in Fig. 1, the SKRM model seamlessly integrates four abstraction layers of cyber situation knowledge in an enterprise network. From top to bottom, they are *Workflow Layer*, *App/Service Layer*, *Operating System Layer* and *Instruction Layer*. As the layer goes down, more detailed and technical information is presented, from business process to the computer semantics and instructions, all at different granularities.

The abstraction layers are by essence the horizontal stovepipes of the enterprise network situation knowledge. Hence, they are abstracted by categorizing isolated situation knowledge, in terms of different levels of computer and information system semantics as well as corresponding expertise, of an enterprise network. In specific, workflow systems are popular to ensure the correctness of daily business/mission processes. Dependencies between a particular application and corresponding multi-services are invaluable to ensure the stability and efficiency of the application [5]. OS object (file or process) dependencies are useful to backward or forward track the intrusion propagation. Dynamic instruction taint tracking is powerful to enable fine-grained intrusion harm analysis. Based on an extensive literature exploration, the Workflow Layer, App/Service Layer, Operating System Layer and Instruction Layer are proposed as important layers in the SKRM. The roles of each of these four abstraction layers are explained below.

SKRM is not a simply mapping from situation knowledge to a set of abstraction layers shown in Fig. 1. Rather, it is a combination of two transformative ideas, interconnecting the perception level elements into comprehension level awareness.

1) Every abstraction layer must cover the entire data network. The layer must integrate and interconnect our awareness of all the network assets (hosts, file systems, memory, and networking) at that layer. This idea is to break vertical stovepipes (e.g., compartments).
2) Cross-layer analysis is the "soul" of SKRM. That is, the cross-layer relationships (e.g., mapping, translation, and semantics bridging) are captured by SKRM. This idea is to break horizontal stovepipes (e.g., abstraction levels).

The following are the main features of the SKRM model:

- Each abstraction layer generates a graph, and each graph covers the entire enterprise network;
- Cross-layer relationships are captured. The individual graphs are interconnected to become a graph stack;
- The graph stack enables both inter-compartment diagnosis and cross-layer analysis;
- Each abstraction layer is a view of the same network from a different perspective and thus at a different granularity;
- Isolated perception that is gained at different layers/granularities is integrated into a more comprehensive, scalable system to support higher levels of SA, namely comprehension and projection.

B. SKRM Layers

1) Workflow Layer

Workflow management is the primary technology for organizations to perform their daily business processes [30]. A workflow is composed of several essential tasks in order to complete a business process. These tasks are arranged in a specific order and dependent on each other. If the workflow management system is compromised, the attacker can forge or corrupt the tasks and data in the workflow by malicious injection or modification. The corrupted workflow may behave abnormally, such as changing the execution paths. Organizations should have a consistent and reliable execution path of workflow to function well. Therefore, we first propose a Workflow Layer in SKRM to capture the business/mission processes within an enterprise.

- **Definition 1 (Workflow Layer):**
  The graph of Workflow Layer can be represented by a directed graph *G(V, E)*, where *V* is the set of nodes (tasks) and *E* is the set of directed edges (immediate precedence relations). If $(t_i, t_j) \in E$, then (t_i, t_j) is a directed edge pointed from task t_i to task t_j, and t_j should be executed subsequently to t_i. The directed edges derive the *data and control dependency* relationships among tasks according to their definitions in [6]. A workflow *G(V, E)* has a start node with 0-indegree, and some end nodes with 0-outdegree. Any path from the start node to the end node is an *execution path*.

The Workflow Layer shown in Fig. 1 could be referred to as an example. In the example, the workflow is composed of 7 tasks for a business process. As examples of dependency relations, t_3, t_4, t_5 are all control dependent on t_2 and t_4 is data dependent on t_2. These tasks can be executed following different orders, and thus form different execution paths.

2) App/Service Layer

As defined in [14], workflows are broken into several individual "block tasks" to execute, and each block task is then resolved as a sub-workflow into a set of "tasks". That is, the execution of workflows ultimately concludes in the execution of tasks, which further depend on the proper function of specific application software. Moreover, according to Chen et al. [5], the functionality, performance and reliability of a particular application may rely on multiple pre-requisite services, spanning many relevant hosts or other components in the network. Hence, we propose an App/Service Layer in SKRM to capture the applications and services on which the execution of workflows depends.

- **Definition 2 (App/Service Layer)**:
  The graph of App/Service Layer can be represented by a directed graph $G(V, E_1, E_2)$, where V is the set of nodes (applications or services), E_1 is the set of uni-directional edges (dependencies), and E_2 is the set of bi-directional edges (network connections). Service nodes are denoted as a three-tuple (*ip, port, protocol*). If $(A_i, S_j) \in E_1$, then (A_i, S_j) is a *dependency* relation from application A_i to service S_j. If $(S_m, S_n) \in E_2$, then (S_m, S_n) is a *network connection* between service S_m and service S_n.

For example, as illustrated on the App/Service Layer of Fig. 1, the tasks from t_1 to t_6 of the above workflow will all leverage the application called "*Avactis Server*", in order to run the web-shop. This application further depends on four services in the network: web service (*httpd*), database service (*mysqld*), DNS service, 3rd party web service (for hotel reservations, etc.). As a result of such dependencies to achieve application functionality, the web service (*httpd*) has network connections to reach the other three services.

3) Operating System Layer

After locating the applications and services to execute the workflow, further exploration is required inside the host. Therefore, we introduce an Operating System Layer into SKRM to build an OS-level dependency graph.

As shown on the Operating System Layer in Fig. 1, OS-level entities can be classified into two kinds of objects: processes denoted by rectangle and files by ellipse. Both process objects (running code) and file objects (files in storage and memory) exist in this layer, as well as "sockets for managing network connections, pipes for doing IPC and many device interfaces" [29].

- **Definition 3 (Operating System Layer)**:
  The dependency graph at the OS Layer can be specified by a directed graph $G(V, E)$, where V is the set of nodes, and E is the set of directed edges which means immediate dependency relations. Each node is a system object (mainly a process or a file inside a system). If $(N_A, N_B) \in E$, then node N_A is influencing N_B in a certain way which we can call as *dependency* [15], represented as edges in the graph. Dependency rules between objects are defined in [15] and summarized in [29].

4) Instruction Layer

Fine-grained intrusion impact diagnosis may identify missed intrusions at the process-file level intrusion analysis and help "sweep out" victim taints later [31]. An Instruction Layer is proposed in SKRM to specify and correlate the memory cells, disk sectors, registers, kernel address space, and other devices. A graph at the Instruction Layer can be generated based on mapping each instruction flow to the corresponding system objects so as to bridge the semantic gaps between OS Layer and Instruction Layer [13, 31]. At this layer, the dynamic taint analysis semantics [19] could be applied. The Instruction Layer of Fig. 1 illustrates such a graph.

- **Definition 4 (Instruction Layer)**:
  A graph of the Instruction Layer can also be specified by a directed graph $G(V, E)$, where V is the set of nodes, and E is the set of directed edges which mean direct *data or control dependence*. A node is an instruction, register, or memory cell. If $(N_A, N_B) \in E$, then node N_B is data or control dependent on node N_A.

IV. SKRM Graph Stack Generation

The above-identified abstraction layers are actually horizontal stovepipes and the compartments (tasks, services, hosts, OS-level objects, instruction-level objects) on the same layer are vertical stovepipes. Such compartmentalization isolates the SA based on these individual situation knowledge collectors. We propose the following solutions of inter-compartment and cross-layer interconnection, to respectively break the vertical and horizontal stovepipes.

A. Inter-compartment Interconnection

The inter-compartment interconnection is actually the process of generating the network-wide graph at the corresponding abstraction layer.

1) Workflow Design and Workflow Mining

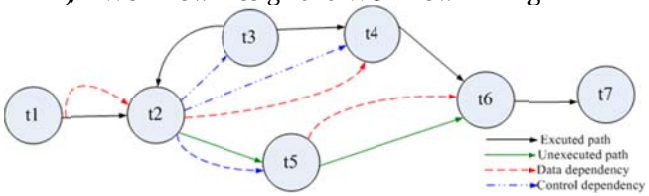

Fig. 2 The graph of Workflow Layer

There are two ways to generate the Workflow Layer. Primarily, workflow can be designed and pre-specified by business managers since the function of a workflow is to achieve a defined business outcome by performing a set of logically related tasks. A professional business manager can describe the specific units of work that the business needs to perform (e.g., a web or database service) and record them as tasks on the Workflow Layer. Alternatively, we can also generate the Workflow Layer with a method called workflow mining[1, 7, 9, 18]. The actual workflows can be extracted by applying data mining technology to the workflow log, which contains information about the workflow process as it is actually being executed. Fig. 2 illustrates our graph of Workflow Layer, applying the first method to the same web-shop business scenario as in [30]. The graph helps us recognize two different execution paths existing in the scenario - P_1: $t_1t_2t_3t_4t_6t_7$ for non-members and P_2: $t_1t_2t_5t_6t_7$ for members. We call them non-member service path and member service path respectively.

2) Service Dependency Discovery

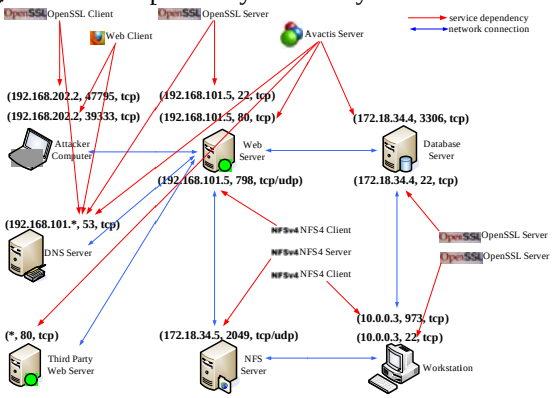

Fig. 3 The graph of App/Service Layer

The graph generation at the App/Service Layer relies on the discovery of service dependencies. Exploiting human expert knowledge is straight forward, but does not scale with the number of applications/services in the enterprise [5]. As alternatives,

several automated discovery of dependencies have been proposed in the field. For examples, [4, 14] derive service dependencies from network traffic patterns, while Chen et al. [5] discover service dependencies based on an observation that "the traffic delay distribution between dependent services often exhibits typical spikes that reflect the underlying delay for using or providing these services". Our graph generation of App/Service Layer (shown in Fig. 3) adopts the scheme presented in [5]. Service dependencies (for dependency) and network connections (for reachability) are differentiated with red and blue colors.

3) Inter-host OS Level Dependency Tracking

According to [8-9], (recent) system call audit logs can be used to determine the dependency relation type between two OS level objects. Hence, following such "dependency rules", system calls can be resolved to process-file dependency relations, which will then be merged to become then OS object level dependency graph.

One thing to emphasize is that we extend the single-host OS object dependency graph to incorporate the socket-based communications between programs, by which we can know whether a local program has been influenced by any remote programs through message passing. This way, the individual dependency graphs generated at each host are literally stitched together. Therefore, the Operating System Layer also covers the whole network, rather than only focuses on a single host. This is especially helpful for inter-host intrusion root identification. Fig. 4 is our resulted graph of OS Layer, incorporating such inter-host extensions illustrated with blue dotted arrows. We also use the red color to highlight the infected OS-level objects. Inter-host extensions are demonstrated to help reveal the propagation of infection.

4) Instruction Level Taint Tracking

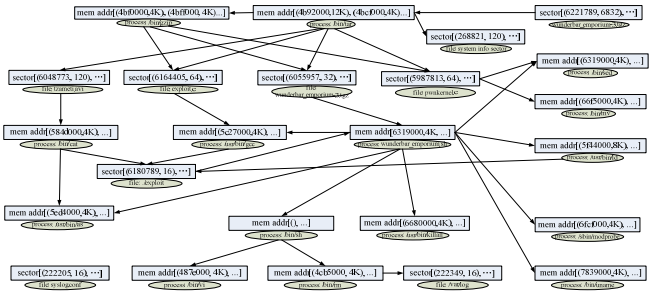

Fig. 5 The graph of Instruction Layer

Two parts of work could be performed at the Instruction Layer, as suggested by [31]. First, fine-grained taint analysis can be applied to generate instruction flow dependency, which contains valuable binary information. Second, cross-layer infection diagnosis can be performed to bridge the "semantic gap" between Instruction Layer and Operating system Layer, by dynamically mapping each instruction flow with corresponding system objects. Therefore, the outcome graph of Instruction Layer shown in Fig. 5 includes two kinds of nodes: instruction-level objects (memory cells or disk sectors represented with rectangles) and mapped OS-level objects (processes or files represented with ellipses).

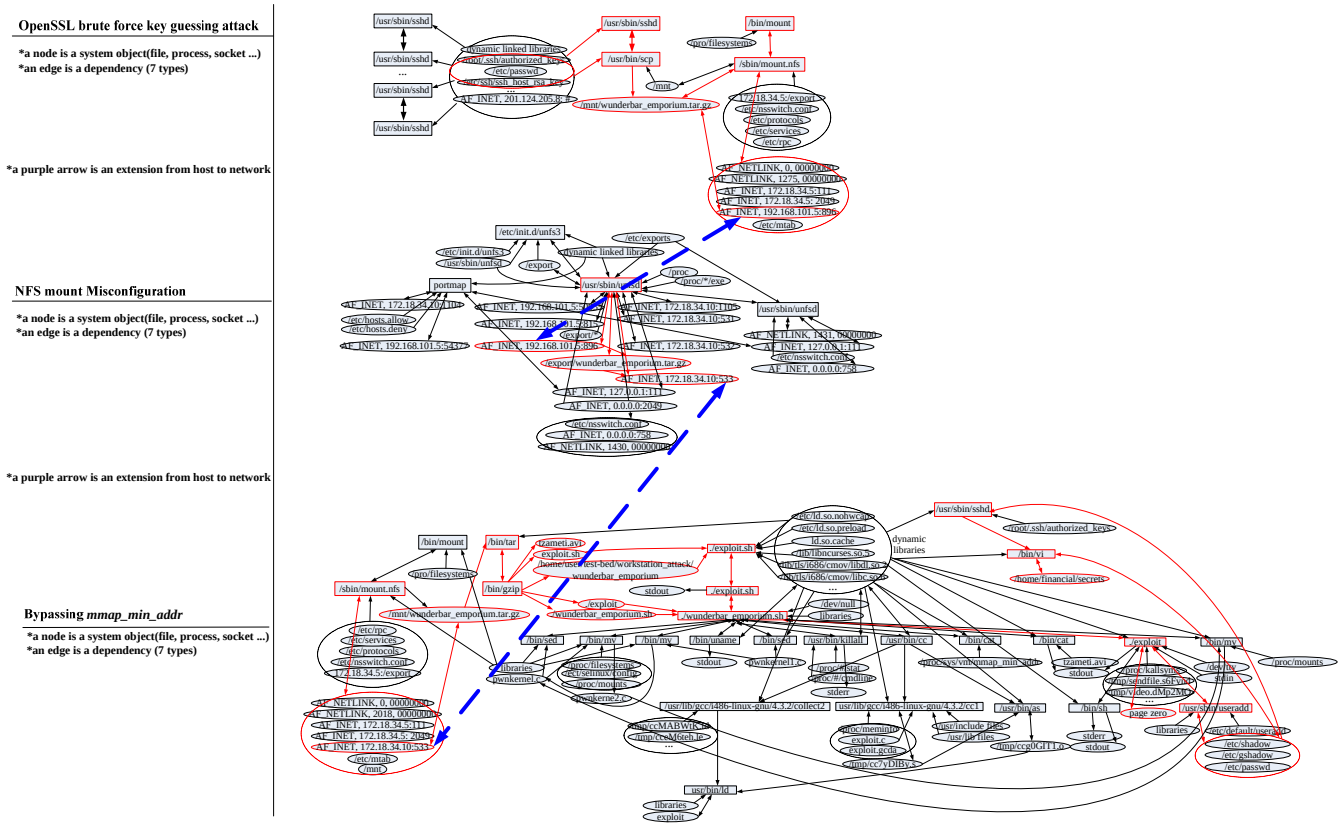

Fig. 4 The graph of Operating System Layer

B. Cross-layer Interconnection

Cross-layer diagnosis is critical for SKRM model, as traversing from one layer to another layer along the edges would lead to expected new information and ultimately a holistic understanding of the whole scenario. However, it cannot be achieved without the fulfillment of cross-layer interconnection. Only with inter-compartment interconnection we still lack the capture of cross-layer relationships that can break horizontal stovepipes.

1) Cross-layer Semantics Bridging

Basically, cross-layer relationships are captured by semantics bridging (specifically, mapping, translation, etc.) in-between the adjacent two abstraction layers of computer and information system semantics. In specific, association between the workflow tasks at Workflow Layer and the particular applications at App/Service Layer can be mined from the network traces with workflow logs, and can be used to create bi-directional mappings between them. The mappings between OS level objects and instruction level objects can be achieved by developing a reconstruction engine such as the one presented in [31]. The purple bi-directional dotted lines between adjacent layers in Fig. 1 illustrate such mappings.

2) Attack Graph Representation and Generation

Specially, we interconnect the App/Service Layer and OS Layer by vertically inserting a dependency Attack Graph between them. This enables the causality representation and tracking between App/Service Layer pre-conditions (network connection, machine configuration and vulnerability information) and OS Layer symptoms/patterns of successful exploits.

- **Definition 5 (dependency Attack Graph)**: The dependency Attack Graph (AG) can be represented with a directed graph *G(V,E)*, where *V* is the set of nodes and *E* is the set of directed edges. There are two kinds of nodes in the attack graph (refer to the attack graph of Fig. 6): derivation nodes (represented with ellipses) and fact nodes. The fact nodes could be further classified into primitive fact nodes (represented with rectangles) and derived fact nodes (represented with diamonds). The directed edges represent the *causality relationships* between the nodes.

In the dependency Attack Graph, one or more fact nodes could serve as the preconditions of a derivation node and cause it to take effect. One or more derivation nodes could further cause a derived fact node to become true. Each derivation node represents an application of an interaction rule given in [28] that yields the derived fact. Let's take our generated attack graph (Fig. 6) for example: *Node 26*, *27* (primitive fact node) and *Node 23* (derived fact node) could cause *Node 22* (derivation node) to take effect, and *Node 22* could further cause *Node 14* (derived fact node) to be valid. Besides, a derived fact node may have different ways to become true.

Fig. 1 illustrates a subset of Fig. 6. Fig. 1 also illustrates the interconnection of the dependency Attack Graph with its adjacent two layers. The conversion from App/Service Layer information (network connection, host configuration, scanned vulnerability) to the primitive nodes in Attack Graph is resulting from the Datalog representation before attack graph generation [28]. The mapping from the derived fact nodes in Attack Graph to the OS Layer intrusion symptoms (such as the system call sequence [10], intrusion pattern, signature, etc.) can be achieved by bi-directional inter-host OS level dependency tracking proposed above, using the OS level instances of host or service configuration as input. For example, the process *"/usr/sbin/sshd"* instantiates *sshd*, and *"/etc/exports"* instantiates *unfsd*. Tracking *"/usr/sbin/sshd"* would reveal the repeated pattern of accessing *sshd*-related processes and files, indicating the occurrence of *Node 14* in the dependency AG.

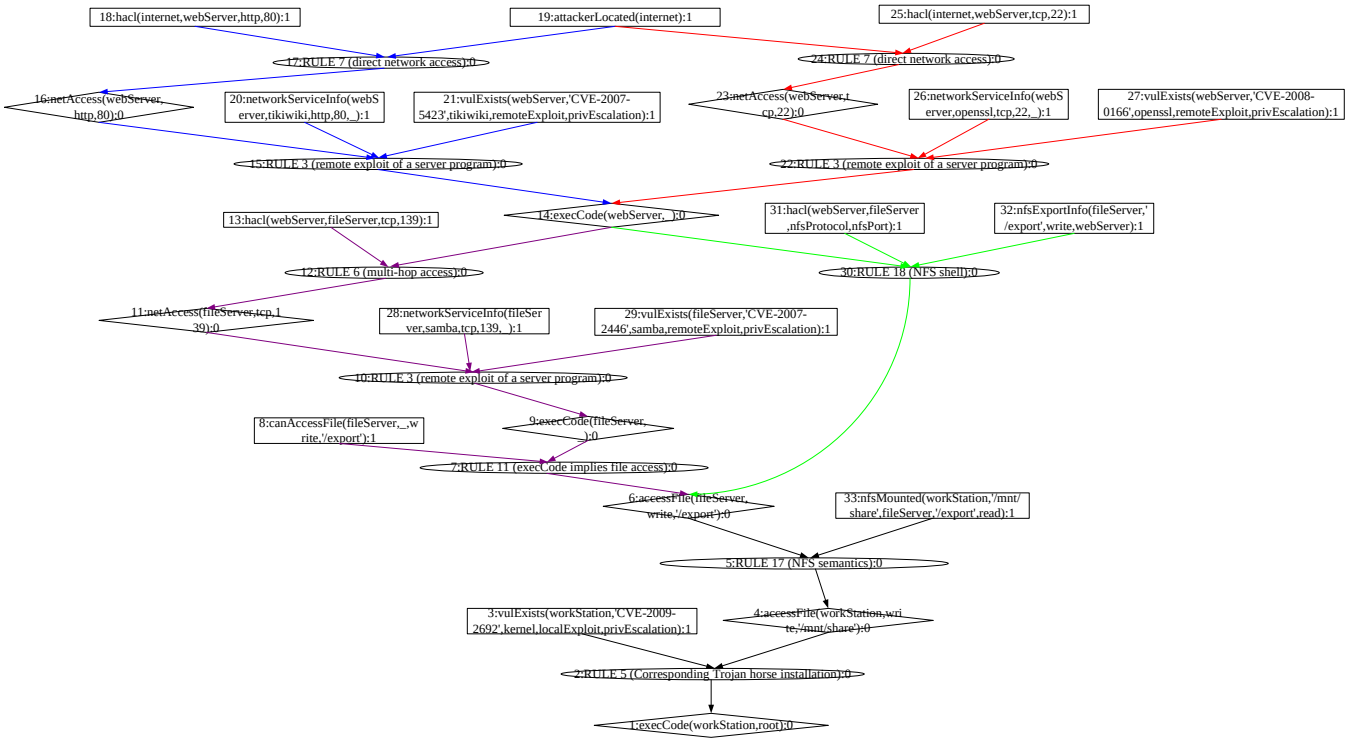

Fig. 6 The dependency Attack Graph

V. Case Study

The security analyst needs to leverage information across different abstraction layers to diagnose an attack and assess its impact in an enterprise network. Business-level symptoms (alerts raised by human managers at high layer) or system level events (alerts provided by security monitoring systems like Snort, tripwire, anti-virus, etc.) are all invaluable to compensate the situation awareness of each other.

Since SKRM is proposed to break stovepipes through cross-layer diagnosis, we present the following case study to demonstrate that the SKRM graph stack is useful to enable capabilities toward holistic perception and comprehension. It is also an illustration of the practical generation of the SKRM graph stack to perform cross-layer analysis.

A. Implementation

To illustrate the application of SKRM framework to cyber security analysis, we implement a web-shop in our test-bed which uses a business scenario similar as the one described in [30]. To observe the network under cyber-attack, we further implement a 3-step attack scenario as in [21, 28] with different vulnerability choices (CVE-2008-0166-OpenSSL brute force key guessing attack, NFS mount misconfiguration, CVE-2009-2692-bypassing mmap_min_addr). The test-bed business and attack scenario is shown in Fig. 7.

In addition, we also deploy intrusion detectors and auditing tools in our web-shop test-bed, such as the Nessus server to scan for the vulnerability and machine information of all the hosts, the MulVAL reasoning engin to generate the attack graph, Snort and Ntop to detect intrusions and monitor the network traffic, and strace to intercept and log system calls. We leverage these situation knowledge collectors to acquire real data for further cross-layer security diagnosis.

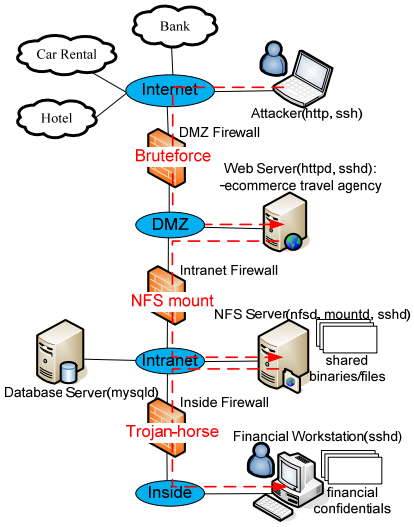

Fig. 7 The test-bed network and attack scenario

B. Capability: Mission Asset Identification and Classification

Usually an obvious intrusion symptom of an enterprise is the business level financial loss. The responsibility of security analysts is to reason over such symptoms so as to identify the exact intrusion root and all the infected mission assets, for better protection and recovery. That is, the capability of mission asset identification and classification is required. As shown in Fig. 8, top-down cross-layer SKRM diagnosis will enable this capability.

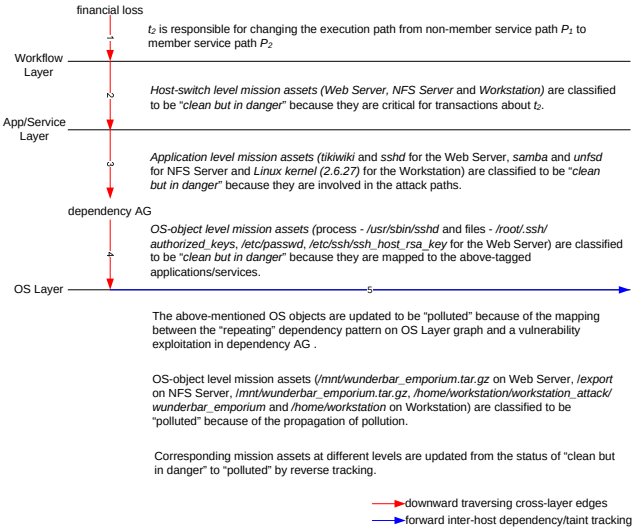

Fig. 8 Mission asset identification and classification

Generally, mission asset identification and prioritization achieves at the identification and classification of host-switch level, application level and OS-object level mission critical assets into such classes as "polluted", "clean but in danger", and "clean and safe". For example, the business managers of the web-shop found the profit much lower than expected. Through analysis on the Workflow Layer (Fig. 2), the security analysts suspected that non-member attackers cheated by getting service from the web-shop via the member service path P_2. According to the control dependence relation in the workflow, they found that task t_2 is responsible for changing the execution path from P_1 to P_2 (step 1). So they tracked down the cross-layer edges between Workflow Layer and App/Service Layer, with particular inspection on task t_2 (step 2). Such cross-layer edges revealed the critical *host-switch level* mission assets involved in transactions about t_2: *Web Server*, *NFS Server* and *Workstation*. Hence, as the most possible attack goals, these assets were tagged into "clean but in danger". The analysts further tracked down the cross-layer edges between App/Service Layer and OS Layer (step 3), and found that there were four possible attack paths in the dependency AG: {*23, 14, 6, 4, 1*}, {*16, 14, 11, 9, 6, 4, 1*}, {*16, 14, 6, 4, 1*} and {*23, 14, 11, 9, 6, 4, 1*}. The four paths all lead to the compromise of Web Server, NFS Server, and Workstation, but exploit vulnerabilities of different applications/services. Fig. 6 differentiates the paths with red, blue, purple and green colors respectively. All the *application level* mission assets involved in the four attack paths were regarded as "clean but in danger": *tikiwiki* and *sshd* for the Web Server, *samba* and *unfsd* for NFS Server and *Linux kernel (2.6.27)* for the Workstation.

The analysts continued to track down the cross-layer edges from dependency AG to OS Layer, and identified fine-grained *OS-object level* mission assets: process - */usr/sbin/sshd* and files - */root/.ssh/authorized_keys*, */etc/passwd*, */etc/ssh/ssh_host_rsa_key* for the Web Server (step 4). These objects were considered as "clean but in danger". The mapping between the "repeating" dependency pattern on OS Layer graph (Fig. 4) and Node 27 in dependency AG (Fig. 6) confirmed the exploitation of CVE-2008-0166. Therefore, the above-mentioned OS objects related to this vulnerability on Web Server could be determined as "polluted".

Further forward dependency tracking on the dependency graph discovered a file named */mnt/wunderbar_emporium.tar.gz* was created and thus "polluted" on the Web Server (step 5). Inter-host OS dependency tracking helped reveal the propagation of such pollution: the file sharing directory */export* on NFS Server was "polluted"; the files or directories named */home/workstation/workstation_attack/wunderbar_emporium*, */mnt/wunderbar_emporium.tar.gz*, and */home/workstation* on Workstation were all "polluted". In a similar way, the memory cells or disk sectors at Instruction Layer corresponding to the system objects could also be classified into these categories.

Through reverse tracking to the upper layers, the status of Web Server and its service sshd, NFS Server and its services unfsd, mountd, Workstation and its service sshd were all updated from "clean but in danger" to "polluted". In a word, through such top-down cross-layer SKRM-based analysis, mission assets at the host-switch level, application/service level and OS-object level could all be identified and further classified into such classes as "polluted", "clean but in danger" and "clean and safe".

C. Capability: Mission Damage and Impact Assessment

Security monitoring systems, such as Snort, tripwire, anti-virus, etc., are effective tools to provide us intrusion alerts, but do not offer us the exact damage and impact. As shown in Fig. 9, the U-shape cross-layer SKRM-enabled analysis helps us to achieve comprehensive damage and impact assessment.

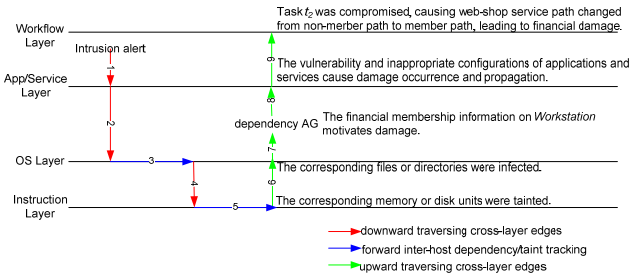

Fig. 9 Mission damage and impact assessment

The scenario begins with a normal status for the web-shop business, but Snort suddenly gives an alert indicating a brute force attack on the *Web Server (sshd)*. The security analyst would like investigate the *Web Server* and start to inspect (scan) its information of applications and services (step 1). The downward traversing cross-layer edges between App/Service Layer and OS Layer reveals the repeated pattern of accessing *sshd*-related processes and files, confirming the occurrence of *Node 14* (indicating successful exploit) in the dependency AG (step 2). Further through the process, the inter-host dependency tracking at the OS Layer identifies the intrusion taint seeds: the file named */mnt/wunderbar_emporium.tar.gz* on the *Web Server*, the directory named */export* on the *NFS Server* and the files or directories named */mnt/wunderbar_emporium.tar.gz*, */home/workstation/workstation_attack/wunderbar_emporium* and */home/workstation* on the *Workstation* (step 3). Using these as input, downward traversing the cross-layer edges between OS Layer and Instruction Layer helps to identify the tainted *memory and disk units* (step 4). The forward inter-host taint tracking at the Instruction Layer located the fine-grained impacts on victim hosts (step 5). At this point, the OS-level and Instruction-level damage has been identified: *the above files and directories were all infected and performing malicious actions at the OS Layer and their memory or disk space were therefore tainted on Instruction Layer*. This triggered the analyst to perform another round of bottom-up analysis to comprehend the damage at other layers. The analyst tracks upward along the cross-layer edges between OS Layer and dependency AG, and determined the *attack path* (step 6 and 7). The attack path, combined with the abnormal behavior on OS Layer, led the analyst to the missing intrusion intent of the attacker: *the financial membership information under the directory named /home/workstation on Workstation is the evidence of the root cause of the damage*. The mappings between dependency AG and App/Service Layer show the specific pre-conditions of the exploits (step 8). *The vulnerabilities and inappropriate configurations at App/Service Layer allow the damage to be caused*. Finally, the analyst tracks upward to the cross-layer relationships between App/Service Layer and Workflow Layer (step 9), and finds that: *task t_2 was compromised, so the web-shop's service path was changed from non-member service path {t_1, t_2, t_3, t_4, t_6, t_7} to the member service path {t_1, t_2, t_5, t_6, t_7} at Workflow Layer and enables significant financial damage to occur*.

In a word, SKRM enables a U-shape cross-layer analysis, as illustrated in Fig. 9, to assess systematic damage and its impact from multi-layer semantics.

VI. Discussion

From the case study above, we identify that SKRM-enabled analytics can exceed the reach of intrusion detection and attack graph analysis, through inter-compartment awareness and cross-layer analysis (top-down, bottom-up, U-shape, etc.). SKRM actually has the potential to enable other capabilities. For example, attack path determination and attack intent identification were also involved in the above U-shape cross-layer diagnosis. The potential capabilities would be explored in future work, including but not limited to:

- U-shape cross-layer diagnosis may help us understand the adversary activity, including the attack path determination and attack intent identification.
- Bottom-up cross-layer analysis may help evaluate mission impact.
- Cross-layer Bayesian networks could be constructed to reason about uncertainty.
- Top-down cross-layer analysis may help us construct mission asset map based on asset classification.
- Comprehensive analysis may help us simulate different strategic mitigation plans.
- Comprehensive analysis may provide insights for intrusion recovery.
- Knowledge representation could be enabled for cognitive engineering.

In addition to the potentials, the current SKRM and SKRM-enabled analytics have some limitations. Although some tools have been developed to generate parts of the SKRM graph stack, the current version of SKRM is still semi-automatic, gaining computer-aided human centric cyber SA. Additional work is still required to evaluate the utility of SKRM in the scale of a real enterprise and more complex scenarios. Our future work will focus on addressing such limitations.

VII. Conclusion

Current cyber SA based on the technologies in intrusion detection and attack graphs lack the capability to address the needs of mission damage and impact assessment and asset identification (and prioritization). This paper proposes a cross-layer Situation Knowledge Reference Model (SKRM) that combines the isolated

situation knowledge collectors and provides comprehension from a larger perspective. It breaks both the vertical and horizontal stovepipes through inter-compartment and cross-abstraction-layer interconnection. Through a concrete case study based on a business scenario in the presence of a 3-step attack, we specifically explain how to generate the graph stack and use it to perform the cross-layer analysis. The results show that SKRM provides expected SA cues beyond intrusion detection and attack graph analysis. We also discuss the potentials and the current limitations of SKRM and SKRM-enabled analysis.

ACKNOWLEDGMENT

We want to thank the anonymous reviewers for their valuable comments, both on the model and language, that helped shape the final version. This research was partially supported by ARO W911NF-09-1-0525 (MURI), NSF CNS-0905131, NSF CNS-0916469, and NSF CNS-1223710.

REFERENCES

[1] Agrawal, R. et al. 1998. Mining process models from workflow logs. *Advances in Database Technology-EDBT'98*. (1998), 467–483.

[2] ArcSight. HP Enterprise Security.: 2012. *http://www.hpenterprisesecurity.com/*.

[3] Axelsson, S. 2000. *Intrusion detection systems: A survey and taxonomy*. Technical report.

[4] Bahl, P. et al. 2007. Towards highly reliable enterprise network services via inference of multi-level dependencies. *ACM SIGCOMM Computer Communication Review* (2007), 13–24.

[5] Chen, X. et al. 2008. Automating network application dependency discovery: Experiences, limitations, and new solutions. *Proceedings of the 8th USENIX conference on Operating systems design and implementation* (2008), 117–130.

[6] Czerwinski, S.E. et al. 1999. An architecture for a secure service discovery service. *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking* (1999), 24–35.

[7] Van Der Aalst, W.M.P. et al. 2003. Workflow mining: a survey of issues and approaches. *Data & Knowledge Engineering*. 47, 2 (2003), 237–267.

[8] Endsley, M.R. 1995. Toward a theory of situation awareness in dynamic systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*. 37, 1 (1995), 32–64.

[9] Gaaloul, W. et al. 2005. Mining workflow patterns through event-data analysis. *Applications and the Internet Workshops, 2005. Saint Workshops 2005. The 2005 Symposium on* (2005), 226–229.

[10] Hofmeyr, S.A. et al. 1998. Intrusion detection using sequences of system calls. *Journal of Computer Security*. 6, 3 (Sep. 1998), 151.

[11] Jajodia, S. et al. 2005. Topological analysis of network attack vulnerability. *Managing Cyber Threats*. (2005), 247–266.

[12] Jason H. Li, Peng Liu. 2010. Cyber Security Analysis and Situational Awareness: Theory and Practice. Tutorial of MILCOM.

[13] Jiang, X. et al. 2010. Stealthy malware detection and monitoring through VMM-based "out-of-the-box" semantic view reconstruction. *ACM Transactions on Information and System Security (TISSEC)*. 13, 2 (2010), 12.

[14] Kandula, S. et al. 2008. What's going on?: learning communication rules in edge networks. *ACM SIGCOMM Computer Communication Review*. 38, 4 (2008), 87–98.

[15] King, S.T. and Chen, P.M. 2003. Backtracking intrusions. *ACM SIGOPS Operating Systems Review* (2003), 223–236.

[16] McAfee, A. and Wagonfeld, A. 2004. Business intelligence software at SYSCO. *Harvard Business School*. 19, (2004).

[17] McGuinness, B. and Foy, L. 2000. A subjective measure of SA: the Crew Awareness Rating Scale (CARS). *Proceedings of the first human performance, situation awareness, and automation conference, Savannah, Georgia* (2000).

[18] De Medeiros, A. et al. 2003. Workflow mining: Current status and future directions. *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*. (2003), 389–406.

[19] Newsome, J. and Song, D. 2005. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. (2005).

[20] Ntop: 2012. *http://www.ntop.org/*.

[21] Ou, X. et al. 2005. MulVAL: A logic-based network security analyzer. *Proceedings of the 14th conference on USENIX Security Symposium-Volume 14* (2005), 8–8.

[22] Paxson, V. 1999. Bro: A system for detecting network intruders in real-time. *Computer networks*. 31, 23 (1999), 2435–2463.

[23] Qualys Guard. Qualys Secure.: 2012. *http://www.qualys.com/*.

[24] Sawilla, R. and Ou, X. 2008. Identifying critical attack assets in dependency attack graphs. *Computer Security-ESORICS 2008*. (2008), 18–34.

[25] Sheyner, O. et al. 2002. Automated generation and analysis of attack graphs. *Proceedings of IEEE Symposium on Security and Privacy (2002)*, 273–284.

[26] Snort: 2012. *http://www.snort.org/*.

[27] Wireshark: 2012. *http://www.wireshark.org/*.

[28] Xinming, O. et al. 2006. A scalable approach to attack graph generation. (2006), 336–345.

[29] Xiong, X. et al. 2009. SHELF: Preserving Business Continuity and Availability in an Intrusion Recovery System. *Computer Security Applications Conference, 2009. ACSAC'09. Annual* (2009), 484–493.

[30] Yu, M. et al. 2004. Self-healing workflow systems under attacks. *Proceedings of 24th International Conference on Distributed Computing Systems (2004)*, 418–425.

[31] Zhang, S. et al. 2010. Cross-layer comprehensive intrusion harm analysis for production workload server systems. *Proceedings of the 26th Annual Computer Security Applications Conference* (New York, NY, USA, 2010), 297–306.

[32] Zonouz, S.A. et al. 2011. Managing business health in the presence of malicious attacks. *Proceedings of the 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops* (2011), 9–14.