CloudSEC: A Cloud Architecture for Composing Collaborative Security Services

Jia Xu, Jia Yan, Liang He, Purui Su
State Key Laboratory of Information Security,
Institute of Software, Chinese Academy of Sciences,
Beijing, China
{xujia04, yanj, heliangliang, supurui}@is.iscas.ac.cn

Dengguo Feng
State Key Laboratory of Information Security,
Graduate Unversity of Chinese Academy of Sciences,
Beijing, China
feng@is.iscas.ac.cn

***Abstract*—Massive Internet invasions implemented through the distributed platform fabricated by rapid diffusion of malwares, has become a significant issue in network security. We argue that the notion of "Collaborative Security" is an emerging trend in resisting distributed attacks originated from malware. Therefore, this paper proposes a new architecture: CloudSEC, for composing collaborative security-related services in clouds, such as correlated intrusion analysis, anti-spam, anti-DDOS, automated malware detection and containment. CloudSEC is modeled as a dynamic peer-to-peer overlay hierarchy with three types of top-down architectural components. Based on this architecture, both data distribution and task scheduling overlays can be simultaneously implemented in a loosely coupled fashion, which can efficiently retrieve data resources from heterogeneous network security facilities, and harness distributed collection of computational resources to process data-intensive tasks. Hence, CloudSEC endues the network security infrastructure with the capability of dynamic adaptation and collaboration on an inter-organizational scale. The results of preliminary evaluation demonstrate that, CloudSEC not only delivers a sample service of distributed intrusion correlation with high scalability and robustness, but also achieves remarkable effectiveness in data sharing and task scheduling.**

Keywords-*cloud architecture; collaborative security; peer-to-peer overlay; distributed data sharing; self-adaptive task scheduling*

V. INTRODUCTION

In recent years, the spread of Internet-scale malware such as worms[21] and botnets[4, 11] has resulted in various distributed intrusions and attacks, which poses a great threat to both critical infrastructure network and organizational Intranet. For example, in June 2004, one of the key DNS servers of the Internet infrastructure provider Akamai was subjected to a DDOS attack originated from botnet, which consequently terminated services of several well-known websites including Microsoft, Google and Yahoo.

Current prevalent practice for safeguarding against the threat from large-scale malware spread mainly relies on the straightforward combination of autonomous security facilities within each single organization, such as firewalls, network intrusion detection systems (NIDS), and anti-malware systems. However, it lacks the ability to rapidly and exactly identify, classify, and respond to large-scale attacks in a distributed and comprehensive manner. We argue that "collaborative security" is an inevitable trend in order to correlate the information obtained from numerous isolated organizations. In this motif, we present CloudSEC, an overlay-based architecture for composing in-cloud collaborative security services such as distributed intrusion correlation, cooperative DDOS and spam filtering, as well as collaborative malware analysis and containment. CloudSEC provides the foundation for incorporating data resources of heterogeneous security facilities across the network into a global perspective, which is of vital importance in detecting and defending against the threats on an inter-organizational scale. Consequently, we summarize the contributions of this paper as follows:

- We propose a novel architecture modeled as a dynamic hierarchy of peer-to-peer overlays, which facilitate to organize adaptive and scalable collaboration upon network security infrastructure on an inter-organizational scale.
- We propose an approach for efficiently sharing structured data collected from diverse security facilities in the Distributed Hash Table (DHT), using a group of XML decomposing and restructuring procedures. Moreover, we propose an application-specific mechanism loosely coupled with the data sharing approach, so that a data-intensive computation task can be split into independent small subtasks and scheduled among autonomous peers. In this way, CloudSEC is capable of composing location-transparent collaborative security services in clouds.
- We implement a proof-of-concept prototype for evaluating CloudSEC's architectural and performance characteristics. The initial results indicate that our idea for the composition of in-cloud collaborative security services is practical and effective.

The rest of this paper is organized as following: Our in-cloud approach on collaborative security are provided in Section 2, the pivotal aspects of the CloudSEC architecture including the schemes of data sharing and task scheduling are demonstrated in Section 3, the specifications of a proof-of-concept implementation of CloudSEC platform as well as preliminary evaluation results are described in Section 4, finally, the research's conclusions and future work are discussed in Section 5.

978-0-7695-4302-4/10 $26.00 © 2010 IEEE
DOI 10.1109/CloudCom.2010.16

II. APPROACH

This paper advocates CloudSEC, an open platform for facilitating the deployment and delivery of a variety of collaborative security services in clouds. CloudeSEC uses a peer-to-peer overlay hierarchy to allow services to be composed into dynamically scheduled tasks with adequate data and computation resource provision. Our approach is characterized by two elementary facets. First, the challenge of online analyzing a great deal of data, such as malicious network traffic and security alerts, can be addressed by utilizing widely spread idle analysis engines in a collaborative way. Second, the detection and protection capabilities currently provided by autonomous security facilities in organizational unit can be more effectively organized as in-cloud services.

A. P2P Based Collaborative Security

Single organization has limited resources to detect and respond to Internet-scale threats. Consequently, the major benefit of a collaborative security is an improved view of global attack activities. The peer-to-peer sharing of information collected from various security facilities across the network is the key facet of such a collaborative approach.

Previous work on collaborative security has focused mainly on distributed intrusion detection. DOMINO[29] is an overlay architecture that utilizes the Chord[24] protocol to support exchange of IDS data between administrative domains. Worminator[13] is a peer-to-peer system that can recognizes attacks in a distributed manner, and claims that their work addresses the challenge of the tradeoff between latency and bandwidth used for exchanging information.

Without being confined to IDS correlation, our work aims to provide extensible collaborative security by integrating heterogeneous detection engines. Therefore, we present a comprehensive approach to cope with the correlation of potential volume of distributed data. We present a data sharing scheme to store structured messages in DHT and provide a globally accessible query interface. A decentralized task scheduling scheme is further proposed to split data correlation into independent subtasks and schedule them with high cost-effectiveness in terms of bandwidth and processing power.

B. Security as In-cloud Service

The recent development of cloud computing has a few key characteristics, including massive scalability, resource manageability and on-demand service delivery[9], which potentially facilitates security service deployment. There is currently a strong trend toward moving services from end hosts and centralized servers into network cloud. For example, in-cloud anti-virus services[7, 18, 19] are explored to identify malwares by multiple executable analysis engines, provably providing better detection accuracy, improving deployment and management. Besides, in-cloud email[15, 22] and HTTP[14, 20] filtering systems are already feasible and adopted as an additional layer of security for enterprise networks. While these approaches mainly contribute to delivering one special-purpose security service that is remotely accessible by users, our work concentrates on generalized characteristics of composing collaborative security services in clouds. Our objective is to fabricate an open platform to facilitate deployment and delivery of custom security services conforming to some restriction conditions. Notably, MOSES[23] attempts to provide an architecture for composing network security services in the form of overlay networks, mostly from which the idea of our paper is inspired.

The core of the proposed approach is moving the analysis and correlation of generalized network alerts from centralized systems into the network cloud. CloudSEC consists of a great number of autonomous agents, while each agent runs multiple lightweight processes to communicate with peripheral security facilities such as intrusion detection systems, network-wide traffic filters(DDOS, spam, etc.), as well as malware behavior analysis engines(worms, bots, etc). First, these agents provide in-cloud data storage functionality, which allows each peripheral security facility to upload crucial network alerts and analysis reports to its corresponding agent, in order to make them be shared in a distributed manner. Second, these agents can be dynamically organized into application-specific task scheduling overlays to perform data correlation and global analysis, which will ensure the provision of on-demand computation resource. CloudSEC strives to integrate heterogeneous security facilities into a self-adaptive system, which enables in-cloud security services to be delivered on-demand with necessary data storage and computation power.

III. ARCHITECTURE

A. CloudSEC Overview

The goal of CloudSEC is to provide a distributed, scalable and robust platform for delivering collaborative security services to their respective users. Towards this end, CloudSEC includes a diverse collection of autonomous peers that dynamically form a P2P overlay hierarchy, as shown in Figure 1.

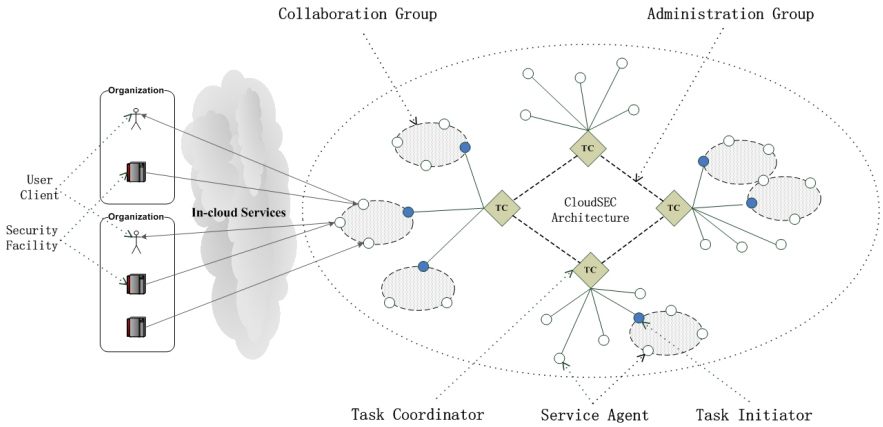

Figure 1. CloudSEC architectural overview

A CloudSEC System is comprised of three sets of components: administration group, collaboration groups and peripheral entities. All communication between the nodes of administration group and collaboration group is encrypted. We will describe each of these in the following sections.

1) Administration group. Administration Group is the kernel component of the CloudSEC architecture, which is an overlay organized by those peers called task coordinator. Each task coordinator is in charge of a dynamic administrative domain, which contains a collection of autonomous security agents. Task coordinators are responsible for making security policy decisions, managing collaboration tasks as well as distributing analytical results across multiple administrative domains. Hence, their scalability and availability is fundamental to the resilience of CloudSEC.

Overlay networks have been proven to be highly resilient to turbulence and demonstrated excellent ability to deliver messages during large-scale network failures[5]. In addition, in order to enhance robustness of the architecture, heartbeat messages are checked to ensure quick detection of infrastructural churn resulting from security agent's joining and leaving, and we have implemented an integrity maintenance mechanism to deal with the partition and regrouping of administrative domains in the case of task coordinator failure.

2) Collaboration groups. Each collaboration group is organized to compose a certain security service by carrying out a collaborative task. CloudSEC harnesses the distributed nature of cloud computing to enable dynamic and strategic organizing of overlay networks in proximity to the service users. Therefore, collaboration groups are the vital components of CloudSEC's "Providing Services on Demand" characteristic. They are composed of a group of dynamically clustering security agents, each covering a wide IP space by offering a set of universal access points to heterogeneous security facilities and user clients, enabling them to supply and consume data resources spontaneously. We establish both data distribution and task scheduling overlays upon collaboration groups. Their composition guarantees that large-scale collaborative security services can be delivered on demand by efficient sharing of data storage and computation power across the boundary of organizational topology.

The first security agent joining a collaboration group is called task initiator, which is the direct correspondent between the collaboration group and its task coordinator. It obtains collaborative tasks and instructions from task coordinator, and in turn reports back task status and final results. In this way, collaboration groups are preserving hierarchical attributes toward the administration group, such that every security agent has the potential to efficiently route all messages within the entire CloudSEC architecture, simply through the forwarding mechanism of task coordinators.

3) Peripheral entities. Peripheral entity is a functionality abstraction summarizing all service contributors and users. These nodes don't implement the CloudSEC protocol, and have no access authority to any architectural overlay. Rather, these nodes could install various heterogeneous security facilities and client software, and simply supply and consume data resources through a combination of push and pull mechanism. Extensions to provide support for CloudSEC would be implemented as plug-ins for these systems.

B. Data Distribution Overlay

Both administration group and collaboration groups have implemented the data distribution overlay, which facilitates the storage and retrieval of XML data in the structured P2P networks. Therefore, CloudSEC could maintain a local and global view of network security activities within desired scopes. Furthermore, with the capability of decomposing and restructuring arbitrary XML fragments in a distributed manner, CloudSEC can divide up a resource-intensive data correlation task into a set of smaller subtasks, and scatter them in a collaboration group to finish the task cooperatively.

1) CloudSEC message framework. In order to maximize interoperability and extensibility, all types of messages exchanged by the CloudSEC protocol are represented in XML format. We have developed a message framework to classifying the basic information required to support the entire architecture, principally by extending the schema proposed in the IDMEF (Intrusion Detection Message Exchange Format) draft[12]. Our framework inherits two message types from the IDMEF (alerts and heartbeats), and provides for additional message types. The six message categories in CloudSEC are as follows 1) Alerts 2) Heartbeats 3) Report Messages 4) Query Messages 5) Update Messages 6) Protocol Messages. While the original alerts of IDMEF are basically designed to modify intrusion detection data, our message framework aims to cover a wider range of heterogeneous security facilities for the purpose of interacting with CloudSEC.

2) Data storage and retrieval approach. One of the fundamental aspects of collaborative security as in-cloud service is to store and retrieve arbitrary data in a distributed manner. Our approach to deal with this problem is to employ a group of decomposing and restructuring procedures to manage unified XML data on top of DHT. Our work adopts the classical Chord[24] protocol as the underlying DHT.

XML[27] is a meta-language for marking up semi-structured data. XML data is characterized by a canonical nested tag structure, which makes it prone to be transformed to an XML tree by means of syntax analysis. Furthermore, in order to completely represent the set of possible path expressions contained in certain XML DTD, we have introduced the path-prefix tree, which is essentially the same as the DataGuides[10]. Thus, we do not give a detailed description on how to construct it. For every message type in CloudSEC message framework, its path-prefix tree should be predefined as input parameter of the following procedures.

To query an arbitrary fragment in XML data, XML Path Language (XPath)[28] is utilized. An XPath expression is a sequence of location steps separated by the symbol "/", and a location step consists of an axis, a node test, and an optional predicates. In accordance with the practical application requirements of CloudSEC, our work merely adopts a subset of XPath, which includes three major types of location step: child axis (/), descendant axes (//) and predicates ([]).

Procedure Ⅰ (Distributed data storage): First of all, for given XML data, we extract all leaf nodes from its XML tree, which possibly include textual nodes, attributive nodes as well as empty nodes. We adopt the following five-tuple array[26] to represent the leaf node:

$$(PID, DID, pe, tag, value),$$

where *PID* represents the DHT key of the peer from which the given XML data is originated, *DID* represents the unique identifier of the given XML data on its original peer, *pe* represents the absolute path expression from the root of XML tree to the node, *tag* is a Dewey order[25] representing structural information of the node, and *value* is the textual value of the node, which can be left empty.

All leaf nodes will be put into a DHT using its node name as the key. An XML tree is constructed based on the alert message, from which two textual nodes and one attributive node are extracted and put into a DHT according to their keys.

Procedure Ⅱ (Retrieval of a leaf node): Assuming *r* is an XPath query message of the following form:

$$|s_1|s_2|...|s_n,$$

where s_i is the node name of each location step, s_n is a leaf node, and "|" represents axis, including child axis (/) and descendant axes (//). Predicates ([]) will be discussed later. The procedure of retrieving a leaf node is as follows:

(1) Query all matching five-tuple arrays from the DHT using the last element in the message *r* (leaf node s_n) as the key.

(2) Omit those records, whose *pe* elements mismatch the content of message *r*.

Procedure Ⅲ (Retrieval of a subtree): Retrieving a subtree whose root is an internal node requires recombination of several XML fragments. The *tag* element in each five-tuple array preserves necessary structural information for such a restructuring process.

Similarly, assuming *r* is an XPath query message of the following form:

$$|s_1|s_2|...|s_n,$$

where s_i is the node name of each location step, and "|" represents axis, including child axis (/) and descendant axes (//).The procedure of retrieving a subtree is as follows:

(1) Traverse the path-prefix tree attached to the DTD of the message *r*, if the last element s_n in message *r* has no subnode, just invoke Procedure Ⅱ with the message *r* as input parameter, otherwise go to step (2).

(2) For every subnode c_i of node s_n, construct a new query message $r_i = |s_1|s_2|...|s_n|c_i$, and then do a nested call to step (1).

(3) Collect five-tuple arrays returned by every invocation of Procedure Ⅱ, use tag element to determine relative position of siblings, and then restructure a subtree with the node s_n at its root.

Procedure Ⅳ (Conditional retrieval): In an XPath expression, the predicates ([]) are used to indicate conditional retrieval of arbitrary XML fragment. To cop with this much more complicated problem, we make further efforts to develop a two-phrase retrieving procedure. This time, we take a specific example of query message *r*:

$$r = /Alert[@messageid="001"]/Source.$$

The procedure of conditional retrieving is as follows:

(1) Extract all predicates ([]) from the original message *r*, and then split it into two parts: $r_1 = /Alert/@messageid$ and $r_2 = /Alert/Source$.

(2) Phrase 1 (Query for Condition): Invoke Procedure Ⅱ with the message r_1 as input parameter, and then retain

the returned five-tuple arrays whose *value* elements are "*001*".

(3) Phrase 2 (Query for Target): Invoke Procedure III with the message r_2 as input parameter, and exclude those five-tuple arrays whose (*PID, DID*) pairs don't exist in the results of Phrase 1.

Thus, incorporated with the data distribution overlay based on XML decomposing and restructuring procedures in DHT, CloudSEC is capable of providing a globally accessible query interface for structured data within the entire CloudSEC architecture, regardless of location dispersion of data sources.

C. Task Scheduling Overlay

To compose an in-cloud security service, such as correlated intrusion analysis, usually requires massive computational resources, which is very likely to exceed the capability of a single security agent. Therefore, it is necessary to deploy a task scheduling overlay in collaboration groups, which is able to dynamically combine a large number of security agents together in order to provide the illusion of a single system with on-demand computing power.

We believe the essence of collaboration group in CloudSEC more or less resembles Computing Grid[9] in their common objective to provide highly parallel computation for data-intensive tasks. However, the collaboration group has several specialized attributes including 1) high autonomy and scalability; 2) tailored to deliver security services on demand to remote users; 3) dynamically configurable services. By contrast, traditional grid scheduling algorithms mainly focus on how to determine an optimal computation schedule based on sufficiently detailed and updated information about the state of available systems, which inevitably bring about excessive overheads. They even become impractical under circumstance of highly scalable infrastructure. Hence, we propose a light-weight, decentralized and self-adaptive approach to organize computational resources, which is especially suited for the application environment of collaboration groups in CloudSEC.

1) Basic scheduling scheme. Our approach adopts a tree overlay analogous to the organic grid[6], since the tree structure is an intuitive way to represent integral paths of subtask distribution and result collection. Our innovation is to implement the tree overlay in a way that is loosely coupled with the data distribution overlay described above. Since the data distribution overlay in CloudSEC has provided a unified interface for accessing arbitrary data, a subtask can be encapsulated as a combination of its execution context and XPath-based descriptor of its required data partition. In this way, it is unnecessary to transfer subtasks with large chunks of physical data, thus the overhead of task scheduling is significantly reduced.

It should be noted that, in this paper, we focus on the scheduling of a specific kind of security service in the network cloud which can be divided into a set of independent and identical subtasks. Further study extending our task scheduling scheme to fit the services composed of synchronously communicating subtasks is still under way.

When a security agent joins a collaboration group with inadequate knowledge about network topology, an ancestor-list is retrieved, containing a random subset of member nodes which have already joined the task scheduling overlay. Each node's ancestor-list is shared upon the data distribution overlay to provide reliable availability. As the tree overlay is being incrementally restructured during the task scheduling progress, the ancestor-list will be continuously updated for each node.

A security service with large-scale computational task is originally dispatched to the task initiator. This task should be split into a set of independent subtasks. One thread is created to process these subtasks in sequence, and the node also keeps monitoring incoming "request" messages from other nodes. If it receives a "request" message and still has any unprocessed subtasks, it will return a "dispatch" message along with a certain number of subtasks. Thus, the requester formally becomes the child of the dispatcher in the tree overlay, and subsequently its ancestor-list is merged with the dispatcher's, and the dispatcher itself is added as the first node into it.

On receiving a "dispatch" message, the child creates a thread to process the dispatched subtasks, while other threads are maintaining communication with its parent or other nodes. When requests are received, the child will dispatch a smaller set of unprocessed subtasks to the requester in the same way. The entire computation propagates in this manner, resulting in the formation of a tree-structured overlay with the task initiator as the root.

During the propagation of subtasks, the formation of circular structures in the tree overlay will lead to rotational scheduling. This situation can be avoided by having each node check its ancestor-list on receiving "request" message. If the requester is already in the ancestor-list, a loop may occur and consequently results in request denial.

When a node finished its own subtasks, it will send requests to the nodes in its ancestor-list sequentially, until it can get a "dispatch" message from one of them. Usually the following dispatch will come from its current parent, according to the updating strategy of ancestor-list, unless the parent encounters an unrecoverable failure of network connection.

The results collection is indeed a reversed process, which follows the paths from every leaf node to the root. Every time a node collects a certain number of results, either computed locally or obtained from its children, it will send them back to its parent. This message can be combined with a "request" message for subsequent unprocessed subtasks.

2) Dynamic adaptation scheme. The collaboration group gradually forms a tree overlay out of initial chaos. However, the basic scheduling scheme ends up with the situation that inefficient nodes have the same probability to be assigned subtasks as efficient ones. It is desirable to

place the best-performing nodes to be closest to the root, so as to enhance overall effectiveness of task scheduling. To address this challenge, we further develop a dynamic adaptation scheme for making the tree topology evolve to dynamically adapt to the real-time conditions. This adaptation process is triggered by the fluctuation of run-time computation performance of each node's children.

In CloudSEC, the performance metric of security agent is defined in terms of perceived throughput, which theoretically is a function of both communication bandwidth and computation throughput. Since no initial knowledge of any node is available, we can only rely on passive feedback from child nodes to continuously acquire their performance in the runtime.

Each node should also maintain a child-list and an obsolete-list. The child-list keeps a record of its active children, which are ranked from high to low by the performance. The performance metric is application-dependent and is approximately evaluated on the basis of the average time-interval for sending back a specific quantity of results. Thus the ranking in the child-list actually indicates the performance of the entire subtree with the child node as its root.

The obsolete-list contains former children of the node, which have either failed to send in results since the last dispatching or utterly lost connection. Besides, when the size of the child-list exceeds a specific number, the least efficient child will be transferred to obsolete-list. Nodes are not released from the obsolete-list until a predefined period of time elapses, during which requests from them will be ignored.

The dynamic adaptation may occur, when a node periodically informs its parent of its best-performing child. If the grandchild isn't in the parent's obsolete-list, the parent then adds the grandchild to its child-list. The node then instructs its child to directly contact its parent. Every successful recommendation brings about a partial restructuring of the tree overlay, and the subtree with the child node at its root is promoted to a higher level. This process aims to move nodes with higher throughput toward the root.

There is an alternative situation. When a node moves its least efficient child *lec* to its obsolete-list, it also replaces the ancestor-list of *lec* with its child-list in reverse order. Since the first node that *lec* is going to contact is its second slowest child, the tree overlay will be kept balanced as much as possible. This process aims to move nodes with lower throughput toward the leaves.

IV. Implementation and Evaluation

A. CloudSEC Implementation

To explore and validate the feasibility of our idea, we implemented a proof-of-concept CloudSEC prototype with the support of JXTA[1] framework, which is a set of open source peer-to-peer protocols generalizing fundamental principles of communication and collaboration among connected devices within a virtual overlay network. We choose JXTA by virtue of its clean conceptual design, its implementation independence as well as its flexible open source infrastructure.

On top of JXTA, we established a layer of middleware composed of four components, including task coordinator module, security agent module, data distribution module and task scheduling module. The first two modules provide CloudSEC architectural features for the peers of task coordinator and security agent respectively, and are separately deployed on individual machines. The latter two modules constitute virtual overlays to provide data sharing service and task scheduling service respectively, and are deployed together with a task coordinator module or a security agent module. Notably, the design of security agent module is completely pluggable and parallel. In order for different collaborative security services to be delivered simultaneously in a configurable manner, each service is composed into a task encapsulating its execution context and XPath-based descriptor of its required data. These four components are developed using JXTA J2SE Libraries (JXSE) v2.5, and leverage MySQL 5.1 for underlying querying databases. All messages exchanged among these peers are protected by mutually authenticated SSL/TLS integrated in the JXTA communication channel.

We also implemented a light-weight plug-in for various security facilities and user clients to format exchanging messages and interact with security agent, through a push and pull mechanism. This plug-in is developed using Java SE Development Kit (JDK) 6 to provide a solution across a variety of platforms such as Windows, Linux and FreeBSD.

B. Preliminary Results

The test bed for our prototype is constructed using the virtualization technology. We install VMware ESXi software on a HP ProLiant sever with four-way four-core Intel Xeon 2.13GHz processor and 32GB memory. VMware ESXi supports infrastructure level virtualization of up to 128 machines on a single server host. Each peer in the CloudSEC prototype is running on one of these virtual machines. Furthermore, the virtual networking capability of VMware ESXi allows us to build complicated network environment for simulation purposes.

Upon this virtualized test bed, we have conducted a series of elementary evaluations, mainly focusing on some crucial architectural features of the CloudSEC prototype. In our evaluation scheme, a distributed IDS correlation module is designed as a sample security service, which is based on the root cause clustering [3] and the attack graph distance[16, 17] algorithms. The attack graph is used to model underlying dependencies among a certain number of attacker exploits and network vulnerabilities. Since every attack scenario can be constructed independently by correlating network security events to its corresponding attack graph, we can split the entire distributed IDS correlation service on the basis of each single attack graph. To be more specific, as the starting point of the sample service, a task initiator is assigned a set of attack graphs,

and then it dispatches them as subtasks to its child nodes. The scheduling of attach graphs proceeds along the paths from the root down to leaf nodes in the tree overlay, so that this resource-intensive service is composed in a collaborative way. We deploy a group of Snort[2] IDS instances in several network segments of the test bed, in order to generate distributed network alerts triggered by the replay of DEFCON 17 CtF packets[8]. Each Snort instance is designated to a security agent, which aggregates large quantities of alert messages into a modest number of security events and makes them available within the entire collaboration group via the data distribution overlay. The preliminary evaluation results of the CloudSEC prototype are summarized in the following sections.

1) Feasibility and robustness. We have evaluated the feasibility and robustness of CloudSEC architecture by observing its capability of adaptive recovery from two major types of peer failure scenarios.

- By simulating the breakdown of a task coordinator, we confirm that the security agents originally attached to it can discover this situation through heartbeat signals, and can contact other active task coordinators separately for registration. Besides, the redundant storage characteristic[24] of data distribution overlay makes it practical to partially recover the status of failing peer.
- By simulating the breakdown of several security agents, we confirm that the collaborative group involving failure peers can discover this situation through a response time-out and can restructure the task scheduling overlay to exclude these peers. Moreover, by further simulating automatic recovery from failure, we confirm that these security agents can rejoin the collaboration group by the instruction of task coordinator.

2) Data sharing performance. We have conducted a series of experiments to confirm the performance of the data distribution overlay. Two datasets are adopted, which consist of alert messages in the IDMEF format, generated by Snort system, and the sizes are 5MB and 50MB respectively. We choose three typical XPath expressions as benchmark queries, including a query with pure child axis (/), a query with descendant axes (//) and a query with predicates ([]). It's a reasonable assumption that peers could exchange every single message within a sufficiently short time period. Therefore, we alternatively measured the performance of queries by simply comparing the number of intermediate messages required to process each query, so that the inherent performance deviation between virtualized and physical environments can be utterly ignored.

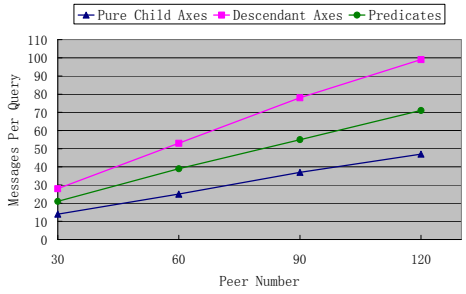

Figure 2. Query performance on 5MB alerts

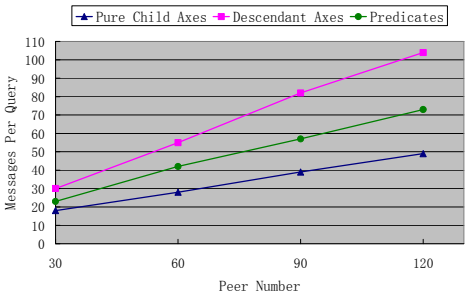

Figure 3. Query performance on 50MB alerts

In Figure 2 and 3, the horizontal and vertical axes show the number of peers and the number of intermediate messages, respectively. We conclude that 1) the query performance turns out to be in approximately linear decline as the peer number grows; 2) the query performance has no dramatic decline, even if the query complexity increases by containing descendant axes (//) or predicates ([]); 3) the variation of dataset size has no significant impact on the query performance. These features indicate that our data storage and retrieval approach is adequate for sharing structured messages in the distributed architecture of CloudSEC.

3) Task scheduling effectiveness. The experiment to evaluate the effectiveness of task scheduling overlay was conducted upon a cluster of 12 security agents running on virtual machines. In order to obtain heterogeneity in performance, we divided security agents into high, medium, low throughput categories by manipulating their hardware configuration to yield differential processor speed and memory size. The distributed IDS correlation task was to match a number of network security events against 150 duplicated attack graphs. Each subtask was to match security events against one attack graph. Attack graphs flow down the tree overlay whereas results flow up to the root. We took three snapshots of the topology of the tree overlay to demonstrate two explicit phrases of the experiment, which are depicted in Figure 4 and Figure 5 respectively.

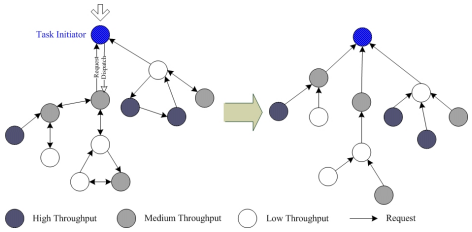

Figure 4. Basic task scheduling

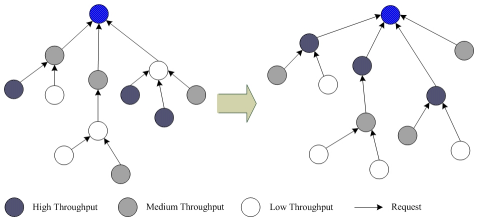

Figure 5. Dynamic adaption

As shown in Figure 4, all peers were initially organized randomly, due to the inconsistency of their original ancestor-lists. The dotted arrows indicate the directions in which request messages were sent. In the first phrase, we conducted the basic task scheduling scheme with dynamic adaptation disabled. The result shows that peers were organized into a stable tree overlay. Nevertheless, different throughput categories were still randomly distributed in this topology.

In the second phrase, we pushed forward the experiment by enabling dynamic adaption. The tree overlay began to evaluate each peer's performance for possible relocation. Figure 5 represents the result of this restructuring process. The final topology shows that fast peers are kept near the root while slow ones are moved towards leaf nodes, which improves the overall effectiveness of task scheduling.

V. Conclusions and Future Work

In this paper, we present and evaluate CloudSEC, a scalable, effective and robust architecture for composing collaborative security-related services in clouds. CloudSEC is designed to unite a group of autonomous peers into P2P overlays hierarchy, consisting of: 1) task coordinators organized in the core administration group, 2) security agents dynamically organized in collaboration groups associated with each task coordinator, 3) peripheral entities, which abstract the functionality of heterogeneous security facilities and user clients widely spread in the network.

One of the key contributions of this paper is introducing a comprehensive approach to combine data distribution and task scheduling overlays in a loosely coupled manner. It constitutes the foundation of CloudSEC, which ensures that in-cloud security services would be delivered to on-demand users with necessary data storage and computational power.

Our initial evaluation of CloudSEC is based on a proof-of-concept prototype, which is deployed in a virtualized, highly configurable test bed. We choose a distributed IDS correlation task as the sample security service. The preliminary results demonstrate that CloudSEC architecture is practical and fault-tolerant, and our proposed data sharing and task scheduling approaches are adequate to fulfill the quality requirements of CloudSEC.

We intend to pursue future work in several directions. First, we plan to extend the proposed task scheduling scheme to be applicable to synchronously communicating subtasks, so that more categories of security services could be composed upon CloudSEC. Secondly, we plan to conduct further evaluations to optimize CloudSEC topology maintenance protocol, and to improve efficiency of both data sharing and task scheduling schemes. We also plan to develop more examples of envisioned security services for CloudSEC, such as cooperative spam and DDOS filtering. Finally, as we emphasize at the beginning, CloudSEC has potential to motivate case studies of large-scale malware. We plan to explore the possibility of developing countermeasures against prevalent malware based on the CloudSEC architecture.

Acknowledgment

This work is funded by National Natural Science Foundation of China under Grant NO. 60703076 and NO. 61073179, and Hi-tech Research and Development Program of China (863 Program) under Grant NO. 2009AA01Z435.

References

[1] "JXTA Community Projects," web page, https://jxta.dev.java.net/, 2007.

[2] "The Snort Project," web page, http://www.snort.org/, 2007.

[3] S. O. Al-Mamory and H. Zhang, "Intrusion detection alarms reduction using root cause analysis and clustering," *Computer Communications*, vol. 32, pp. 419-430, February 2009.

[4] The Honeynet Project and The Honeynet Research Alliance, "Know your enemy: Tracking botnets, using honeynets to learn more about bots," *The Honeynet Project Whitepaper*, web page, http://www.honeynet.org/papers/bots/, May 2005.

[5] D. Anderson, H. Balakrishnan, and F. Kaashoek, "Resilient overlay networks," in *Proceedings of 18th ACM S-ymposium on Operating System Principles*, Banff, Alberta, Canada, pp. 131-145, October 2001.

[6] A. J. Chakravarti and G. Baumgartner, "The organic grid: Self-organizing computation on a peer-to-peer network," in *Proceedings of the First International Conference on Autonomic Computing*, pp. 96-103, 2004.

[7] Cloudmark, "Cloudmark authority antivirus," web page, http://www.cloudmark.com, 2007.

[8] Def con, "Capture the Flag (CtF) packet captures," web page, http://www.defcon.org/, 2009.

[9] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," *IEEE Grid Computing Environments*, December 2008.

[10] R. Goldman and J. Widom, "DataGuides: Enabling query formulation and optimization in semistructured databases," in *Proceedings of the 23rd International Conference on Very Large Data Bases*, pp. 436-445, 1997.

[11] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon, "Peer-to-peer botnets: Overview and case study," in *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, Cambridge, USA, pp. 1-1, 2007.

[12] Intrusion Detection Working Group, "Intrusion Detection Message Exchange Format," web page, http://www.ietf.org/rfc/rfc4765.txt, 2007.

[13] M. E. Locasto, J. J. Parekh, A. D. Keromytis, and S. J. Stolfo, "Towards collaborative security and P2P intrusion detection," in *Proceedings of the 2005 IEEE Workshop on Information Assurance and Security*, United States Military Academy, West Point, NY, 2005.

[14] A. Moshchuk, T. Bragin, D. Deville, S. D. Gribble, and H. M. Levy, "Spyproxy: Execution-based detection of malicious web content," in *Proceedings of the 16th USENIX Security Symposium*, August 2007.

[15] Barracuda Networks, "Barracuda spam firewall," web page, http://www.barracudanetwork.com, 2007.

[16] S. Noel, M. Jacobs, P. Kalapa, and S. Jajodia, "Multiple coordinated views for network attack graphs," in *Proceedings of the IEEE Workshops on Visualization for Computer Security*, pp.12-12, 2005.

[17] S. Noel, E. Robertson, and S. Jajodia, "Correlating intrusion events and building attack scenarios through attack graph distances," in *Proceedings of the 20th Anual Computer Security Applications Conference*, pp.350-359, 2004.

[18] J. Oberheide, E. Cooke, and F. Jahania, "CloudAV: N-version antivirus in the network cloud," in *Proceedings of the 17th USENIX Security Symposium*, pp. 91-106 2008.

[19] J. Oberheide, E. Cooke, and F. Jahania, "Rethinking antivirus: Executable analysis in the network cloud," in *2nd USENIX Workshop on Hot Topics in Security (HotSec 2007)*, August 2007.

[20] N. Provos, "Spybye," web page, http://www.monkey.org/~provos/spybye, 2007.

[21] S. Qing and W. Wen, "A survey and trends on internet worms," *Computers & Security*, vol. 24, pp. 334-346, June 2005.

[22] S.Sidiroglou, J.Ioannidis, A.D.Keromytis, and S.J.Stolfo, "An email worm vaccine architecture," in *Proceedings of the 1st Information Security Practice and Experie-nce Conference (ISPEC)*, pp. 97-108, 2005.

[23] S. Sidiroglou, A. Stavrou, and A. D. Keromytis, "Mediated overlay services (MOSES): Network security as a composable service," in *Proceedings of IEEE Sarnoff Symposium*, Princeton, NJ, USA, 2007.

[24] I. Stoica, R. Morris, and D. Karger, "Chord: A scalable peer-to-peer lookup service for internet application," *ACM SIGCOMM Computer Communication Review*, vol. 31, pp. 149-161, August 2001.

[25] I. Tatarinov, S. D. Viglas, and K. Beyer, "Storing and querying ordered XML using a relational database system," in *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pp. 204-215, 2002.

[26] T. Amagasa, C. Wu, and H. Kitagawa, "Retrieving arbitrary XML fragments from structured peer-to-peer networks," in *Proceedings of the joint 9th Asia-Pacific web and 8th international conference on webage information management conference on Advances in data and web management*, pp. 317-328, 2007.

[27] "W3C: Extensible Markup Language (XML) 1.0 (Fifth Edition) W3C Recommendation," web page, http://www. w3.org/tr/xml, November 2008.

[28] "W3C: XML Path Language (XPath) Version 1.0 W3C Recommendation," web page, http://www.w3.org/TR/xpath/, November 1999.

[29] V. Yegneswaran, P. Barford, and S. Jha, "Global intrusion detection in the DOMINO overlay system," in *Proceedings of Network and Distributed System Security Symposium (NDSS'04)*, San Diego, CA, February 2004.

[30] J. Xu, D. Feng, and P. Su, "Research on network-warning model based on dynamic peer-to-peer hierarchy," *Journal of Computer Research and Development*, vol. 47, pp. 1574-1586, September 2010.