"Meta Cloud Discovery" Model: An Approach to Integrity Monitoring for Cloud-Based Disaster Recovery Planning

Brittany M. Wilbert and Qingzhong Liu

***Abstract*—A structure is required to prevent the malicious code from leaking onto the system. The use of sandboxes has become more advance, allowing for investigators to access malicious code while minimizing the risk of infecting their own machine. This technology is also used to prevent malicious code from compromising vulnerable machines. The use of sandbox technology and techniques can potentially be extended to cloud infrastructures to prevent malicious content from compromising specialized infrastructure such as backups that are used for disaster recovery and business continuity planning. This paper will discuss existing algorithms related to current sandbox technology, and extend the work into the "Meta Cloud Discovery" model, a sandbox integrity-monitoring proposal for disaster recovery. Finally, implementation examples will be discussed as well as future research that would need to be performed to improve the model.**

***Index Terms*—Cloud infrastructure, disaster recovery, integrity monitoring technique, sandbox.**

I. Introduction

As the Internet environment has changed, there has been a constant rise in malicious code and software, with a focus of compromising available machines. This network is now "inhabited by a much larger and more diverse group that includes pranksters, crackers, and business competitors" [1]. As these attacks become more and more sophisticated, the need for reliable technology to limit the impact of these attacks is based on the concept that "an application can do little harm if its access to the underlying operating system is appropriately restricted" [1].

As a result, sandbox technology has been developed to address these concerns. By minimizing the access that is available for applications to operate, other applications on the operating system are protected from potential compromise. Examples of uses of sandboxes is web browser improvements, isolation of exploits to the kernel and the use of honeypots to covertly capture the activity of attackers while allowing the attacker to not realize that their attempts at compromising the system have been minimized. However, the discussion of sandboxing public cloud instances used for disaster recovery is in its development stages.

Many cloud instances use existing policies and controls that are set depending on the implementation of the cloud instance. Although this is a good way to prevent intrusions into the system, it has not been discussed how this data can be audited to ensure that data has not been leaked from these systems, or the integrity of the cloud instance has remained. As a result, there is a potential risk of compromise that may not be located until much later.

To address this concern, an initial framework called "Meta Cloud Discovery" (MCD) has been formulated. MCD is a model framework that is aimed to provide both integrity modeling and assurance that backup data that is sent to a cloud has integrity and non-repudiation throughout its use. This model combines the use of techniques similar to file integrity monitoring and log monitoring to access the state of a disaster recovery instance.

This paper first describes the need for this type of technology by discussing a number of examples of existing sandboxing algorithms and their positives and negatives in relation to the MCD model. Next the model is described which provides the different mechanisms that have been formulated in order to provide both security and functionality to the model. After that, a description of two different implementations of this model, with a public cloud instance and a private cloud instance, is articulated to discuss the benefits and drawbacks this initial model may have on these infrastructures.

Manuscript received May 20, 2013; revised June 26, 2013.

The authors are with the Department of Computer Science at Sam Houston State University, Huntsville, TX 77056 USA (e-mail: bmw005@shsu.edu, qxl005@shsu.edu).

II. Background

The deployment of cloud deployments either as private instances or through the use of public clouds (such as Amazon Web Services) has allowed business to scale down large physical instances to smaller systems while allowing for the extension of their businesses. However, as the result of the use of the cloud other types of fundamental business requirements need to evolve to meet the needs of these new types of infrastructure requirements. An instance of this is the discussion of disaster recovery and what processes are required in order to have an effective and efficient program has been discussed in detail.

File integrity monitoring is one of the solutions that must be implemented in order to provide security to an environment. Monitoring changes to the state of the different software that has been deployed within an environment by using a data 'fingerprint' file is absolutely necessary to provide visibility into potential compromises that have occurred within the environment. However, as a result of the difficult that can emerge as a result of putting into place a file integrity solution, many small businesses forgo this process, and may as a result lose one way to determine when a compromise has occurred and how the system was compromised.

A. Client-Side: Web Browser

Some of the first technologies which sandboxing was

DOI: 10.7763/IJIET.2013.V3.328

developed for were on the web browser itself. An example of this is the use of sandboxing of JavaScript. Agten *et al.* described an example of this in their paper [2]. JSand, their sandboxing framework, uses wrappers, which "consults the security policy to determine whether or not the corresponding operation is permitted" [2]. If a policy is not matched, JSand prevents the activity from activating. This is done while not modifying the browser itself. These policies use a Proxy API 'membrane' to confine JavaScript code from executing outside of the designated Sandbox location, therefore preventing malicious code from activating. Also, this type of implementation, which is independent of the web browser, allows for better portability and backwards compatibility of the software. Its implementation allows for it to be used in multiple products, such as "Google Analytics, Google Maps, and the jQuery library" [2]. This type of detection provides portability of the technology across multiple types of web content, however the independence of this framework may potentially cause vulnerabilities if a web content type does not lend to compatibility to JSand. Although three examples were discussed a further analysis of other web technology would be useful.

Cao *et al.* describe another example of limiting JavaScript activity through the use of a Virtual Browser technology [3] [4]. An initial implementation of this technology by the group was created in 2008. This framework uses virtualization to provide isolation of JavaScript for security. Unlike earlier JavaScript isolation models that used iframes to isolate JavaScript activity, the Virtual Browser adapted strings of JavaScript codes to attach the activity to 'flows' to do evaluations. If trusted code is found, it is allowed to operate in the browser, otherwise the browser uses a set of 'lows to isolate and analyze the activity allowing for the JavaScript code to be avoided or redirected [3].

The later version of Virtual Browser discussed in 2012 was expanded to include the new functions that JavaScript allowed (the 2008 framework only worked for JavaScipt that was currently accepted. This architecture lays on top of the web browser, and not independent of it like JSand. This design still allows for execution of JavaScript code within its sandbox that is then evaluated [3]. Both the 2008 and 2012 versions also included parsing algorithms for HTML and CSS code that is viewed in the web browser [3][4]. The 2008 and extended 20012 versions of the Virtual Browser allow for greater connectivity between the web browser and the Virtual Browser itself by using virtualization technology. Since the virtual browser lies over the web browser it becomes the first line of defense for malicious code. However, from the papers themselves it appears to have difficult handling types of web content, such as Web Sandbox which do not have compatibility with current JavaScript functions [3]. As a result each Virtual Browser version would need to be adapted to allow for communication between unique sandbox and browser environments.

B. Client-Side: Native Client

Another example of browser-based sandbox technology is Native Client. This framework of this client is to isolate x86 native code, which can be run by web browser extensions such as "ActiveX7 and Netscape Plugin Application Programming Interface (NPAPI) allowing native code to be loaded and run as part of a Web application" [5]. As a result, an attacker can compromise a non-sandboxed web browser. Native Client (NaCl) is designed with a collection of components (trusted and untrusted), which are given their own private address space [5]. Communication of the NaCl modules with the browser is done with two different options that are designed to reduce overhead for high volume and frequency communications [5]. Each NaCl module is treated as untrusted code that is then isolated and evaluated. This technology does allow for the ability to detect native code that is very important. However the performance tests that were performed by the authors [5] suggests that additional challenges would need to be overcome in order to continue to decrease the overhead of untrusted native execution on the system. This implementation also doesn't support languages such as Java that are more abundance on the Internet.

C. Server-Side (Website) Protection

Besides securing from the client-side that was discussed in the previous two sections, protection on the server can be just as important. ForceHTTPS allows for the server to enforce strong security policies on the client, in particular asking "the browser to treat HTTPS errors as attacks, not as simple configuration mistakes" [6]. This technique is done in three ways [6]:

1) "Non-HTTPS connections to the site are redirected to HTTPS, preventing contact to the site without TLS."
2) All TLS errors, including self-signed certificates and common-name mismatches, terminate the TLS session."
3) Attempts to embed insecure (non-HTTPS) content into the site fail with network errors."

This methodology is used to address "passive network attackers, active network attackers, and imperfect web developers" by increasing the detecting of unusual activity on the website as well as preventing client-side activity which may result for mishandling improperly written or malicious code. This model also prevents trusted website from leaking data that was resulted from suspicious code from other websites. ForceHTTPS allows for site-controlled defenses that allow for security administrators to conform the policies that are placed.

This framework also allows for rewrite rules to be implemented for client-side permissions to be put in place to prevent unsecure activity such as HTTP and SWF files from compromising secure content [6]. This technology allows for better protection server-side to potential attacks by requesting for strong policies on the client. However implementation of this stronger security model would need to rely on multiple websites to implement ForceHTTPs or another similar framework. Although they do have an alternate solution with a client-side ForceHTTPS structure, implementing either the client or server side versions require "Power" or technically sound users, which can be a barrier for the average user.

D. Wireless

Another location which sandboxing can be used is through wireless. Attackers can use nodes to "capture a node and tamper the applications running, in order to perform different types of attacks" [7]. The framework that is discussed by Zaharis *et al.* uses live forensic monitoring, through sandboxing to detect software tampering while alerting owners of the network of a potential compromise attempt.

This sandbox framework provides a controlled set of

resources that guest processes can run as well as preventing activity such as network access, inspecting host system or reading from an input device [7]. This technique is run in a virtual machine that uses Java objects to isolate each application in use. Each isolated application is then verified for malicious activity [7]. The implementation then evaluates for different activity, such as replay attacks, forgery attacks and other types of tampering of hardware and software [7]. This type of framework is important for locations that allow for wireless access, such as cafes or restaurants that need tampering detection. However, this type of technique, with the potential number of sensors and processing threshold required to operate, may become too cost prohibited for businesses which may want to implement such a technology.

E. Comparisons of Previous Sandbox Methods to MCD

These examples of sandbox technologies have all be sampled in order to construct a model that can be used for the cloud environment. Although a cloud instance does not inherently contain a "client" instance, the requirement for limiting what data is authorized within a cloud backup is very important. The use of JSand's Proxy API 'membrane' as an example on how to create a verification structure is important. The risk of vulnerabilities such as insertions of malicious code or other artifacts into an ecosystem that is purposed to supply a company a secondary source of their data encase of an failure of their primary systems is extremely high. This also needs to be considered because a cloud instance of a backup is inherently connected to the internet where malicious users could compromise the system if it is not harden.

The use of Native-Clients for cloud instances can also be considered. For this model to work for the long-term, the implementation must be able to be lightweight enough to be implemented without limiting the time it takes for a cloud backup instance to sync to the production environment.

Server-side protection for a cloud-based instance is even more important to review. The use of inherited rules that are created in order to minimize insecurity activity, similar to what occurs with "ForceHTTPS" has been introduced into the MCD structure as one of the means to protect what activity is authorized in and out of the cloud instance.

In additional, Zaharis *et al.* monitoring framework provides a base for what a cloud instance could do to provide protection against compromise. The use of objects as a means to verify changes within an environment is extremely useful for cloud instances. Similarly to wireless networks, cloud instances use multiple virtual environments which are combined together in order to provide storage and management capability.

III. Meta Cloud Discovery Model Overview

Meta Cloud Discovery's purpose is to combine the policy and configuration necessities to create a functional cloud instance while introducing means to create a structure to provide verification of integrity and non-repudiation within a cloud instance that is designed to be a part of a disaster recovery planning system. In over to create this, what has been created is the use of "layers". First, an algorithm that serves as the backend for this infrastructure has been initially developed using a type of metadata objects that directly provided relationship to data that is inserted into the cloud environment. In addition, a total of four layers within the model have been identified to serve as a means for verification of activity that occurs within the system. These layers are the following:

1) Metadata Security Layer/Module
2) Integrity Layer
3) Maintenance Layer
4) Reporting Layer

The architecture of this type of design can be reviewed in Fig.1:

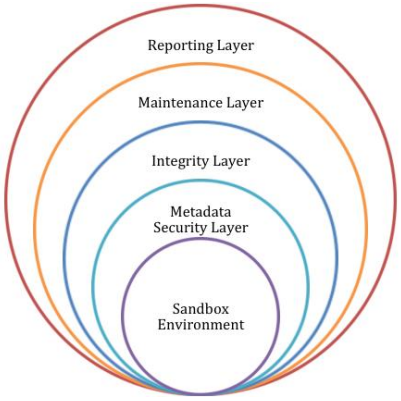


Fig. 1. Meta cloud discovery model design overview.

IV. Model Methodology

The methodology for both the metadata backend components and each "layer" of protection for this data will be discussed in more detail in this section.

A. Metadata Backend Overview

Data is transported from primary data center to backup location. When data enters the backup environment the following occurs:

Algorithm is ran which determines the approximate size of each data cluster. As data is transmitted, cluster is defined to closest X% ending at end of file. Calculation is then made for exact size to thousands of cluster. Cluster is then tagged with the following metadata:

1) Exact size of cluster (block) between selected maximum and minimum block sizes
2) Data of data insertion
3) Name of server instance (if being used in public cloud instance, optional otherwise)
4) Name of company (if public cloud instance, optional otherwise)

Information is hashed using an encryption method of choose (such as MD5 or a stronger algorithm). Although encryption and decryption are important, it must also be aware that this will reduce the amount of time it takes to complete the data transportation, conversion and upload to the sandbox because of the computational requirements for these methods. To mitigate this problem, the results are stored in the Metadata Security Layer, which is enclosed within the policy-enabled sandbox and is used as a means to enable the other layers of the implementations for maintenance, monitoring and reporting. This is done by created an indirect link of the created metadata that is then

stored in Metadata Security Layer.

For larger data sets, such as databases and other instances where data must continue to be placed logically together to prevent failures in synchronization, the block size and minimum and maximum sizes of the instance can be customized to fit those needs. Alternately those instances can be ignored if necessary and placed in the smaller sandbox instances within the policy-enabled sandbox with the maintenance and monitoring layer used to reduce the risk of changes being made to those instances. More about this will be discussed later in the paper.

The basic algorithm used to inset metadata instances into block of works like this block of pseudo-code below:

```
VARIABLE SELECTMIN
VARIABLE SELECT MAX

METHOD blocksize {
blocksize.MB=RANDOM((SELECTMIN,SELECTMAX)}

METHOD metadata {
SUM(blocksize.unique+date.unixtime+serverinstance.hexva
lue+company.hexvalue) }

METHOD decrypt {
DATADECRYPTION METHOD}

blocksize.unique = CALLMETHOD blocksize+0.075%
INSERT blocksize.unique
LOCATE date.unixtime
SELECT serverinstance
CONVERT serverinstance.hexvalue
IF INSTANCE = publiccloud{
SELECT companyname
CONVERT company.hexvalue
} ELSE company.hexvalue=0

CALLMETHOD metadata
datablock.unique = APPEND(block.blocksize+metadata)
NUMBER(ID.unique) = NEW ID datablock.unique
datablock.unique = datablock.unique+ID.unique
INSERT datablock.unique INTO metalayer.security
ENCRYPT datablock.unique

IF report.initiated {
CALL METHOD decrypt(datablock.unique)}
```

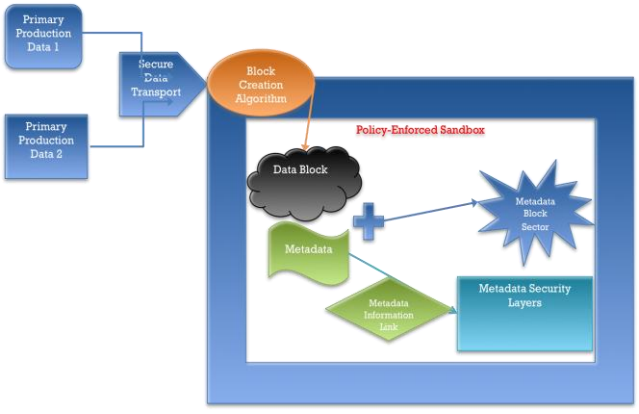


Fig. 2. MCD transportation and blocking configuration design.

Fig. 2 above shows the basic backend design for how the transport and storage process would work within this framework.

B. Protection and Monitoring Layers

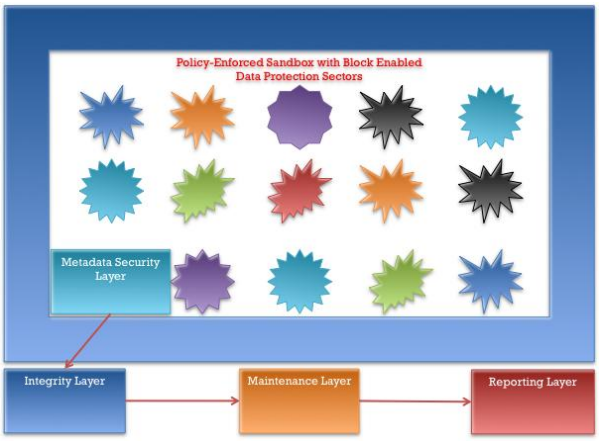


Fig. 3. Overall layer design in relation to sandbox.

Fig. 3 shows how the monitoring layers relate to each other from a higher level in relation to the Metadata backend design. The colored sections are the metadata block sections that were created through the metadata backend after completion of the block insertion into the sandbox clod environment. Each of the protection layers will be described in more detailed in the next four sections.

1) Metadata security layer

As discussed in the previous section, the Metadata Security Layer may be considered as the most important portion of the model. This layer is used extensively as a means to provide integrity protection for the instance.

Within this layer, the metadata indirect links are provided a reference point in which the remaining layers can reference instead of referencing the sandbox directly.

This layer can also have encryption (such as MD5 or another stronger encryption method) enabled. However, encrypting this layer would cause an increase in the amount of time to complete the backup. This is because each time the sandbox is updated or monitored, the layer would need to be decrypted, updated, and then encrypted again to prevent tampering of the resource. As a result, more research into how to best handle this type of additional security and to decrease the time to compete encryption would serve as a way to improve the model.

2) Integrity layer

This layer is used to ensure that the integrity of the sandbox is maintained. Also, this serves as the intermediary point between the Metadata Security Layer and the remaining layers.

The purpose of this layer is to contain the information about the block size set by the company, verify when backup instances have been performed, and ensure that changes are only performed when authorized. This will be maintained by information being stored while serving as the point when integrity checks will be started, stopped and completed. Results from these checks can be referenced in other layers.

3) Maintenance layer

This layer serves as a means for authorized administrative

user accounts to update important settings for the model. This includes the following:

1) Block Size
2) Integrity check scheduling
3) Frequency of backups
4) Authorized users to the system
5) Settings for the sandbox
6) If company name metadata value will be incorporated
7) If server name metadata value will be incorporated.

By minimizing the number of administrative user accounts, this limits the risk of unauthorized users gaining access to the sandbox cloud instance. Since this layer would be set to only allow authorized users to set the permissions that are used by the Integrity layer (and the sandbox itself) it is a point that can be leveraged to limit the locations entry to the backup cloud instances can be used from.

4) Reporting layer

A reporting layer is required to allow for the cloud sandbox infrastructure to be monitored for unusual activity that could occur. To do this, reporting could potentially be created for the following criteria (although not limited to this selection):

1) Integrity checks
2) Monitoring results
3) Results of backups
4) Attempts to access the system with escalated privileges
5) Review of policies and configurations of the sandbox.

This layer would house reporting mechanisms that will allow for reporting to be conducting while limiting the number of users who are provided administrative privileges.

C. Example Implementations

To provide an example on how this model can be utilized, different examples of cloud environments will be used, including a private cloud environment, which is maintained by the company requiring disaster recovery, and a public cloud environment maintained by a third party provider.

1) Private cloud environment

In such an environment, private cloud instances are maintained and monitored by a private company that has established the need for a disaster recovery cloud infrastructure. The model should include the following considerations.

1) Should the company include the company name and/or the server name in the metadata? Since only the company should be storing their own data into the instance, and depending on the size of the instance, these settings may not be necessary in all situations (and decrease storage size).
2) What additional storage requirements are needed? Storage for backups would most likely need to be increased by some amount to fit the additional requirements for the model.
3) What block sizes should be used to create the block sectors? Larger block sectors can increase the computation requirements of the process, and increase the amount of time to complete a backup.
4) What is the Service Level Agreement (SLA) agree to with their customers? Many companies have SLA requirements that need to be met that include the disaster recovery process. A company must consider if this model would significantly change how their SLA should be worded.

Although these considerations have to be made, the advantage of this model is that it can monitor when there are changes to the backup and minimize the chance of the backup being compromised when not in use. This can be leveraged by companies in order to minimize the risk of a secondary disaster occurring during their DR process because the company did not realize that their cloud instance backup has been compromised until too late.

2) Public cloud environment

In addition to the considerations discussed in the Private Cloud Environment example, additional implementation considerations would need to be made for Public Cloud instances. These considerations are below:

1) Public Cloud instances have multiple customers' data within the system. As a result, there is the risk that vulnerability from one customer data set can potentially compromise another business's data. For such an important resource such as a backup instance, would the company want to use a Public Cloud instance as a source for their data?
2) Would the third party provider be able to add this model to their infrastructure?
3) What other measures does the third party company to prevent data leakages from occurring?

The positive of using the MCD model is that it can be used as a remediation tactic to frequently monitor public cloud instances.

V. Future Research and Considerations

Although some of the requirements for improved integrity monitoring for cloud instances, there are future research opportunities that can be done to improve this model. The current model currently is not optimized for efficiently convert the block sizes to include the metadata area.

Also, this type of design leans more to incremental backups where organizations have a large data that is infrequently changed. Is this a fixed point? Can additional, larger data sets use this model? Further research in how this may work for large database instances would determine if this type of model could be implemented.

Since there are many types of cloud instances, unique review of each cloud instance would need to be performed to make sure that configuration and policies enabled on the cloud instance can be converted to sandbox environment. Potentially, the Reporting Layer of model or a mixture of multiple methods to protect these assets can then monitor this.

VI. Conclusion

Integrity monitoring of backups is extremely important to protect potential compromises to these systems. However, as industries transition to potentially using cloud infrastructure to reduce the cost of creating disaster recovery backups for their environments, the need for improved integrity monitoring is even more relevant. Meta Cloud Discovery is an initial model that has been created to attempt to provide a model that may serve as a way to provide integrity monitoring to these systems.

The use of metadata backend, with a security layer model

is used as a means to limit what is inserted and deleted into the systems while monitoring when authorized (and potentially unauthorized) changes to the sandbox are attempted. With the example implementations, there has been shown to be some positives that this model provides. However, further research is required to ensure that this type of model can be used for future instances of cloud environments as well as to stabilize what can be done with using this model with larger infrastructures which have larger variance in their the data that would be needed to be backed up to the cloud environment.

ACKNOWLEDGEMENT

We highly appreciate the support from SHSU research and sponsored program under an Enhancement Research Grant and the support from the Department of Computer Science.

REFERENCES

[1] I. Goldberg, D. Wagner, R. Thomas, and E. Brewer, "A Secure Environment for Untrusted Helper Applications (Confining the Wily Hacker)," *6th USENIX UNIX Security Symp.*, © 1996 USENIX Association.

[2] P. Agten, S. V. Acker, Y. Brondsema, P. H. Phung, L. Desmet, and F. Piessens, "JSand: complete client-side sandboxing of third-party JavaScript without browser modifications." in *Proc. 28th Annual Computer Security Applications Conf.*, doi: 10.1145/2420950.2420952, pp. 1-10.

[3] Y. Cao, Z. Li, V. Rastogi, and Y. Chen, "Virtual browser: a web-level sandbox to secure third-party JavaScript without sacrificing functionality," in *Proc. 17th ACM Conf. on Computer and Communications Security*, © 2010 ACM, doi: 10.1145/1866307.1866387, pp-654-656.

[4] Y. Cao, Z. Li, V. Rastogi, Y. Chen, and X. Wen, "Virtual browser: a virtualized browser to sandbox third-party JavaScripts with enhanced security," in *Proc. 17th ACM Conf. on Computer and Communications Security*, © 2012 ACM, doi: 10.1145/2414456.2414460. pp. 8-9.

[5] B. Yee, D. Sehr, G. Dardyk, J. B. Chen, R. Muth, T. Ormandy, S. Okasaka, N. Narula, and N. Fullagar, "Native Client: a Sandbox for Portable, Untrusted x86 Native Code," *Commun. ACM*, vol. 52, issue 1, pp. 91-99, © Jan. 2010 ACM, doi: 10.1145/1629175.1629203.

[6] C. Jackson and A. Barth, "ForceHttps: protecting high-security web sites from network attacks," in *Proc. 17th Int. Conf. on World Wide Web*, © 2008 ACM, doi: 10.1145/1367497.1367569, pp. 525-534.

[7] A. Zaharis, A. I. Martini, L. Perlepes, G. Stamoulis, and Panayotis Kikiras, "Live Forensics Framework for Wireless Sensor Nodes Using Sandboxing." in *Proc. 6th ACM Workshop on QoS and Security for Wireless and Mobile Networks*, © 2010 ACM, doi: 10.1145/1868630.1868643, pp. 70-77.

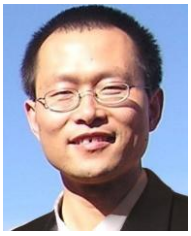
[8] R. Schaefer, "The epistemology of computer security," *Softw. Eng. Notes*, vol. 34, issue 6, pp. 8-10, © Dec. 2009 SIGSOFT, doi: 10.1145/1640162.1655274.

[9] J. Cappos, A. Dadger, J. Rasley, J. Samuel, I. Beschatnikh, C. Barsan, A. Krishnamurthy, and T. Anderson, "Retaining Sandbox Containment Despite Bugs in Privileged Memory-Safe Code," in *Proc. 17th ACM Conf. on Computer and Communications Security*, © 2010 ACM, doi: 10.1145/1866307.1866332, pp. 212-223.

Brittany M. Wilbert is a master's candidate for information assurance and security studying at the Department of Computer Science at Sam Houston State University. She has a bachelor's degree in computer science with an Emphasis in Digital Forensics that was also obtained at Sam Houston State University.

She currently works as an IT Audit and Compliance Analyst at Alert Logic, Inc. in Houston, Texas. Her job duties include working with various departments within the company to research and document improvements to process and procedures performed by the company to follow compliance and audit standards. Her research interests include log management and how to more efficiently incorporate international, federal, and state regulation and compliance standards requirements for corporate, small and medium businesses.

Ms. Wilbert currently is a member of ISACA as well as the Houston, Texas local chapter of ISACA. She is currently holds a GIAC Systems and Network Auditor (GSNA) certification from the Global Information Assurance Certification (GIAC).



Qingzhong Liu is currently an assistant professor in computer science at Sam Houston State University. He obtained his PhD degree in Computer Science from New Mexico Institute of Mining and Technology. His research interest includes information security, digital forensics, multimedia computing and analysis, bioinformatics, pattern recognition, data mining, and computational applications.