The Many Issues of a Human Review Downgrader

Jon Johnson

The Many Issues of a Human Review Downgrader

by Jon Johnson
Date Submitted: January 24, 2005
Version 1.4c
Option 1

ABSTRACT

In the post 9/11 society, we have all been put on heightened alert to the importance of making sure that data is only disseminated to those that have a need or right to the information. Items such as blueprints and delivery schedules now must be looked at as sensitive information [5]. The government and military have always dealt with this problem. They have created information domains, which are various labels denoting the level of sensitivity of data such as top secret (TS) and unclassified [1]. We will refer to the information domain of higher sensitivity as the "high side" of the guard and the other side of the guard as the "low side."

It is logical to assume in the government and military worlds that information would need to be passed between information domains, but the process of passing information from low to high information domains or vice versa has rigorous rules governing how the transfer can occur, when it can occur, by whom, etc. The usual answer to the problem is incorporation of a guard, which sits between the two information domains where transfers occur. Guards can come in an automated form or one that requires human review of the information before it can be released. What follows is a discussion of human review downgraders and the issues surrounding them [1, 12].

INTRODUCTION

A human review downgrader is a guard that deals with data transfers from the high side to low side and requires human intervention to pass information [12]. We discuss general guard topics: intransitive noninterference, the functions of guards, types of guards, guard certification, and the role guards play in Multilevel Security (MLS) and Multiple Independent Levels of Security (MILS) architectures. Then we discuss follow-on topics: sanitation of data, non-repudiation, human certifiers, logging, and implementation issues [1].

GUARD FUNCTION

"A guard is a device used to defend the network boundary by employing the following functions and properties:

- Typically subjected to high degree of assurance in its development.
- Supports fewer services.
- Services are at the application level only.
- May support application data filtering (review).
- May support sanitization of data.
- Typically used to connect networks with differing levels of trust

(provides regrading of data)." [12, pg 6.3-1]

Guards are placed between networks of different information domains, and they are comprised of many hardware components and software programs such as firewalls, sanitation programs, and virus programs [12]. They can either be dual-homed where there is a different network interface card (NIC) for each network connecting to the guard or single-homed which only has one NIC for incoming and outgoing traffic [14].

A guard uses all of its various components to provide services that none of the individual components have the ability to provide. The main functions this paper focuses on are the ability to move information from the high side to the low side, the sanitation ability, and the mating of that with human reviewers. In order for a guard to provide passage of information from the high side to the low side (i.e. to downgrade its sensitivity level), the guard must use automated checks of the information and certification of the data for it to pass through, or a human must review the data running needed tests and sanitation programs that are necessary to certify the information for release. The high expectations for the behavior of the guard are maintained through rigorous certification processes [1, 5, 12]

INTRANSITIVE NONINTERFERENCE

> "The term 'intransitive noninterference' refers to the information flow properties required of systems like downgraders, in which it may be legitimate for information to flow indirectly between two users but not directly." [9, pg 1]

This definition is one of the key points of how a guard operates, that the guard is a "gate keeper" of the information that passes back and forth from the two separate systems. Also, the definition alludes to the fact that communication through a guard is not a direct connection to the other side. To communicate we must first explicitly or implicitly connect to the guard machine, and the guard will make the connection on the other side if it determines such a connection is allowable [1]. This functionality is similar to other network devices such as routers that use network address translation (NAT). All requests to outside networks go through the router, and the router has the ability to send the data to the other side or block the information depending on the rules that it maintains. Also as part of the NAT, the router hides the Internet protocol (IP) address of the sending machine to the other side, which is a function that can be found in guards as well [15].

The topic of intransitive noninterference is the framework for the study of how the interactions of the low and high-level users on a system affect each other and the system. A variety of means to study these interactions exists such as

process algebra, state machines, and trace theory [9].

Process algebra looks at how to mathematically study and define process behavior. Like most algebra studies there are certain base axioms assumed to be true from which all other analysis is derived [10].

State machines on the other hand are involved more heavily in set theory. A language is defined to describe a specified set. This set can be represented in many forms from a compact formula to graphical representations. The goal is to define a finite language such that it encompasses all of the possible states that the guard can function in. With this description it is easier to analyze the correctness of the guard and its ability to maintain separation of the two information domains. Not only is it easier for the designers to deduce flaws in the system, but it would also allow for help in the certification process of the guard. However, the task of creating such a language to describe the guard is a non-trivial problem [2].

Trace theory begins with the idea of trying to define how the various components of a bigger system behave. Then from these components, models can be created. The interactions between the models can be found by analyzing the behaviors of the individual components [11].

The primary goal of all the various methodologies is to try and create a deterministic way to show how the interactions of both sides of the guard occur. Such modeling can be an aid in showing that the guard has a high degree of assurance, which in turn helps in getting the guard certified. Furthermore, the theories that are developed can be applied to the multi-level security (MLS) paradigm, which is discussed later. [1, 9, 12]

TYPES OF GUARDS

Many types of guards exist for the various situations and needs of a system. The types of guards are generally denoted by the way in which they allow information to flow. One group of guards does not require the need to make decisions concerning downgrading data. Some examples include a guard that only allows information to pass from low to high and a guard that passes information from two information domains of the same sensitivity level. The other group does require making a decision concerning whether the data may be released to the low side of the guard. The decision to release the data can be done by automated processes or a combination of automation and human intervention [1, 12].

The ideal in most situations is to have a fully automated guard that does not require human interaction in order to perform its duties. In many cases this ideal is impossible to implement due in part to complexity of the sanitation processes, complexity of the data, and regulations. Where full automation is

impractical a human review component can be incorporated into the guard to facilitate the sanitation of the data and the decision to release the data or not.

A human reviewer has two primary locations to operate. They can be outside of both of the networks that they are responsible for or they can be in the high side network. Normally, when they are placed outside of the networks they are put in the path between both of the connecting guards for the individual networks. This allows the reviewer the ability to intercept all traffic and allow or disallow information. On the other hand if the reviewer is placed in the high side network, then all requests that go to the guard for sending information from that network to the other must first get the approval of the reviewer. [12]

CERTIFICATION

As mentioned in the section on guard function, a guard is "subjected to high degree of assurance in its development" [12, pg 6.3-1]. The realization of this statement comes from the stringent certifications that a guard is subjected to in order to become operational. Gaining certifications is a long and costly endeavor in most cases, but many levels of the certifications exist. A general rule of thumb is that the higher the needed assurance level the more costly and involved the process becomes. Some examples of certifications are "Secret and Below Interoperability (SABI), Director of Central Intelligence Directive (DCID) 6/3 and the Common Criteria (CC)" [1, pg 7]. Another related example is Top Secret and Below Interoperability (TSABI) [1, 12].

ROLE IN MILS/MLS ARCHITECTURE

The multilevel security (MLS) architecture allows for data of different sensitivity levels to be contained on the same system. Multiple independent levels of security (MILS) architecture, on the other hand, segregates the data so that the data in various information domains is separated [1, 8].

One of the major components of the MLS architecture is requiring labels for all data. The concept of labeling is the electronic equivalent of marking a paper file with a sensitivity level. MLS has many system wide requirements that can interfere with its ability to integrate commercial off the shelf (COTS) software as well as government off the shelf (GOTS) products. The need to label all files contained in the system as well as MLS capable operating systems on all computers introduces many demands on interoperability and a cohesive labeling scheme. This type of system can have implementation issues such as "cost, compatibility, and assurance" [8].

The MILS architecture on the other hand does not allow files of different sensitivity levels on the same machine, thus the need for labels only exists for data that is being moved from one information domain to another or for the case that there is data with a sensitivity level different then the predominate type for

that system. This is the type of architecture assumed for use with the human review downgrader discussed in this paper. Under the MILS one side of the guard is maintained at one information domain level and the other side can be at the same or different level. Another benefit of MILS is the easier incorporation of COTS products in system design, which can help in the cost of the overall system [1, 8].

SANITATION

The basic motivating factor for sanitation is the fact that many COTS software packages of today have irregular behavior, and their implementations may not perform in they way we assume. Moreover, with the increasing use of COTS programs, we try to use the programs to perform a function for us that it was not originally designed to accommodate. Therefore, the COTS may perform as it is originally intended to, but with our attempt to add other functionalities we can create erroneous behavior. Sanitation is the cleansing of a document to a point where it no longer contains any information, code, viruses, etc. that could transport information from the high information domain to the lower information domain [1, 8].

Sanitation programs are necessary to allow the file to be released to the low side. One commonly used program that is guilty of keeping data in the file that is not clearly visible to the user is Microsoft Word®. Word® has certain functionality concerning the use of multiple authors and auto save that result in information being kept in the file. Many sanitation programs exist for Microsoft Word® files that take out the hidden data such as file path information, contributors, and deleted data [3].

However, just getting rid of the hidden data is not necessarily enough. Other programs go through the content of the data and look for information that cannot be released to the low side and delete it. The type of program that performs this function is a “dirty word” checker [1, 3, 8].

NON-REPUDIATION

One of the key issues about the transfer of data in today’s world is the ability to know who sent a particular piece of data i.e. non-repudiation. Normally, we use a digital signature on a segment of data to encrypt the data in such a way that when the data is decrypted the receiver can be assured that they know who the sender is. A digital signature makes use of public key cryptography in order to accomplish this task [13].

Another side to the story of non-repudiation is that it is necessary to have some way to know if the data is changed in any way. This is where the concept of a hash comes into play. A hash is a mathematical computation using the information sent as the data to the formulas. Once a hash is computed then this

short data string can be digitally signed. Now a receiver of the data can tell if the data was altered, as well as know if the data is sent from the expected sender. A hash comes in many different forms such as MD5 and SHA1 [13].

So far we have been discussing the abstract idea of digital signatures solving the problem of non-repudiation; however, many major issues make this a much more complex problem. In order to create a digital signature the sender must have a private and public key, and they must distribute the public key to all receivers of information from them. They also need to be able to tell receivers if the private key of the sender is compromised. These larger issues fall under the realm of a public key infrastructure (PKI). Some of the problems in this area include revocation list and key servers, which are non-trivial in nature. [6]

Taking the discussion back to the downgrader, we need a way for a human reviewer to non-reputably certify that they have cleared certain information for release to the lower sensitivity information domain. One possible method of implementation is the use of digital signatures mated with a hash of the data, which can give clues to the data having been modified and ensure they have cleared the data for release [13].

For the scenario where the reviewer is inside the information domain, a guard is able to certify that the appropriate reviewer has verified the information they are about to release and that it has not been modified. Other issues follow this that can be solved based on the needs of the system.

For the case where the reviewer is in between the two guards of the information domains, it is possible for the sending guard to sign the data, the reviewer to verify the data and sign it herself. The receiving guard can then verify that the reviewer certified the information and that it can be released to the low side.

As can clearly be seen there can be multiple entities involved in the certification of the data for release to the low side. Depending on the needs of a given system, it is possible to implement a system that allows for the multiple signatures to stay with the file as it traverses to the low side. This adds a twist similar to how digital certificates function. Each entity in the chain would have to have the ability to verify the correctness of each of the signatures that had accumulated to that point for the given data transfer. Then if any of them were corrupt in any way the default response is to cancel the transfer. In certain situations and depending on the system requirements a method to reacquire a corrected signature may be possible.

Also, the downgrader has to have the ability to interface with the PKI infrastructure. Some of the actions it must perform are the ability to retrieve keys, check for revoked keys, etc. However, due to the complexities involved with the PKI system itself, special care must be taken to ensure the correctness of the interface. For example the downgrader has to be able to ensure it is

connecting to the correct key servers, and then it must determine that the keys it received are up to date and have not been tampered with [6].

HUMAN CERTIFIERS

To this point we have operated under the assumption that the downgrader cannot be fully automated because of the complexity of the verification processes for the decision to release the information; so we introduce humans to review the data. But the introduction of human interaction adds its own risks and vulnerabilities to the system [4, 7].

In the fast paced, hectic lifestyle of today's work environment, many employees can find themselves overworked and distracted. This can lead to bad judgment and hinder the ability of the certifier in making final determinations on the release of the data.

Another common problem that can hinder work done in sensitive information domain environments is the access that may be granted to the certifier to analyze the file. Checks may be able to be run on the file but the certifier may not be able to actually look at the data due to its sensitivity. Also, human certifiers have a great deal of responsibility put on them to behave in a manner consistent with the company or government rules.

However, as history shows there is always a risk that humans may be able to be bribed or coerced to give up sensitive information or allow it to pass through. To mitigate this risk a system of checks and balances such as dual reviewers can be made.

The next issue we must look at is the skill level of the certifier. How skilled a certifier is at running all necessary tests and the ability to analyze the output from those programs could impact the correctness of what data is allowed to pass through the downgrader [4, 7].

LOGGING

Another vital component of the downgrader system is the ability to log all the events. When the human review component of the downgrader is part of the same machine then the logging issue is less complex. The interesting case occurs when the human reviewer is located on a different physical machine. In this case each machine can create its own logs of the events, but then those events have to be time correlated to the events of the other machine in order to recreate the correct sequencing. This implies the introduction of mechanisms that sync the clock between the two machines and a central repository for all the log data.

As mentioned previously there are many complex issues regarding non-

repudiation of the data that is being transferred from the high information domain to the low information domain. Those issues carry over to logging as well. For example logs would need to be made by the guard as to who it believes signed a particular document okay to release. If a chain of signatures exists, each individual needs to be logged. One of the large issues present with this is how does a guard "know" who signed the document. The determination can be made as a part of the PKI, from the guard's retrieval of public keys. They can be given unique identification information that allows the downgrader to say which ID it thought verified the data. But again we revisit the issues surrounding the PKI infrastructure. The downgrader's ability to determine who verified the data then resides in the security revolving around the PKI system [6, 13].

IMPLEMENTATION ISSUES

Under the stipulation that a guard can never release data that is not allowed, then in any situation that a guard or a human reviewer finds any reason to reject the file, the default reaction must be to fail the transfer for the information [12].

The guard has to have its code base and assets protected from modification except by those selected to do so. Another issue is that the guard machine can have both low and high files coming in and out of the same machine (depending on the type of guard and direction of flow). This is the area of MLS verses MILS, where the MLS machine has the ability to handle having data from two information domains.

Also, we must look at the two different implementations for the human review downgrader. For the case of the human reviewer being inside the high side information domain there is a need to protect all program files relevant to the functions of the human certifier. However, since data from two information domains will not reside on the machine, this allows for less stringent data handling requirements.

For the implementation of the downgrader where the human reviewer sits between the two guard machines, we need to have protection of both the programs for the human reviewer and for all the data that passes through because data from two different information domains will exist on the machine. Again we see the need for ideas found in MLS architecture.

Another issue is involved in how the communication between the machine for the guard and the human reviewer takes place. A guard must know of a request for a file and have an ability to be signaled when the correct number of reviewers has verified the file.

Then on the reviewer machine a file has to be handled in such a way as to prevent two reviewers from accessing the same file at the same time for the purposes of sanitization. The issue here is that a reviewer may have to do

multiple saves of a file as various sanitation programs are run. Then if another reviewer accessed the file under normal operations, they have read access and they are dealing with a copy. If they tried to do a save of the file, we would have inconsistent data.

The data itself also contains some issues. Under the MLS framework every data file would need to be labeled as to its sensitivity level. Beyond the issue of sensitivity level there is also a need to have a standard form for packaging such as how to include the signatures other information necessary to tell the guard machine and the human reviewer what they need to know about the file [12].

EXAMPLE FLOW

The file request is sent from the low side to the high side. The request has to pass through the guard to ensure that it is a well-formed request and passes all the tests that must be run. Then what happens next will vary depending on the specific implementation of the system. One of the tasks that must be performed is the notification of the human reviewer that a request for a file has been made. Once notified the reviewer can then retrieve the file and begin the process of making a decision to downgrade the file or not. If multiple reviewers are necessary then the reviewer can notify the appropriate individuals.

After all the decisions are made, if it is a failed request then no action is taken, but when the file is cleared to be released then the guard machine must retrieve the file. The guard at this point must run all of its own automated checks on the file to ensure the file passes for release. Also, the guard must verify that all the appropriate parties have valid signatures signifying the file is releasable. As mentioned in the *Logging* section previously, all of these actions on the various machines must be properly logged in order to create an audit trail in case an error is made. The audit trail allows for discovery of what went wrong [12].

CONCLUSION

The complexity of the guard increases depending on the functionality. A human review downgrader is one of the most complex implementations of guard architecture due to the introduction of issues such as non-repudiation, human reviewers, logging, sanitation, and certification. The need for a guard not only exists between information domains but can also be utilized anywhere a system requirement is to be able to control the flow of data from the physical to application layer [8]. The importance of guards is going to increase as the need for communication increases. Also, as the world shifts more to global alliances in both the commercial and governmental arenas, a need will exist to be able to share data among the participating partners in well-defined, predictable ways.

References

1 – Maney, Charles. “Security Issues When Data Traverses Information Domains: Do Guards Effectively Address the Problem?” November 11, 2004. <http://www.giac.org/practical/GSEC/Charles_Maney_GSEC.pdf>.

2 - Lewis, Harry R. and Papadimitriou Christos H. Elements of the Theory of Computation, 2nd ed. New Jersey: Prentice-Hall, 1998. pp 47.

3 - Kernan, Deborah. “Hidden Data in Electronic Documents.” November 11, 2004. <http://www.giac.org/practical/GSEC/Deborah_Kernan_GSEC.pdf>.

4 - Swaim, Kevin. “Humans Are Always The Weakest Link.” January 6, 2005. <http://www.giac.org/practical/GSEC/Kevin_Swaim_GSEC.pdf>.

5 - Breeding, Alexander J. “Sensitive but Unclassified Information: A Threat to Physical Security.” November 11, 2004. <http://www.giac.org/practical/GSEC/Alexander_Breeding_GSEC.pdf>.

6 - Angela Keith. “Common issues in PKI implementations – climbing the ‘Slope of Enlightenment’.” November 10, 2004. <http://www.giac.org/practical/GSEC/Angela_Keith_GSEC.pdf>.

7 - Eric Padilla. “The Human Factor.” January 6, 2005. <http://www.giac.org/practical/Eric_Padilla_GSEC.doc>.

8 – Smith, Rick. “The Challenge of Multilevel Security.” January 4, 2005. <http://www.smat.us/crypto/docs/BH%20MLS%2006a.pdf>.

9 – Roscoe, A.W. and Goldsmith M.H. “What is intransitive noninterference?” January 4, 2005. <http://www.cs.cornell.edu/andru/cs711/2003fa/reading/roscoe99what.pdf>.

10 – Glabbeek, Rob van. “Process Algebra.” January 23, 2005. <http://theory.stanford.edu/~rvg/process.html>.

11 - Passerone, Roberto. “Algebraic Trace Theory.” January 23, 2005. <http://www-cad.eecs.berkeley.edu/Respep/Research/hsc/class.F03/ee249/lectures/Lec12.pdf>.

12 - The Information Assurance Technical Framework Forum (IATFF). Chap. 6.3. Release 3.1. September 13, 2004. < http://www.iatf.net/framework_docs/version-3_1/file_serve.cfm?chapter=ch06s3.pdf>.

13 – Viega, John, Messier, Matt, and Chandra, Pravir. Network Security with OpenSSL. Cambridge: O'Reilly®, 2002. pg 9.

14 – SANS Security Reading Room. "Proxies and Packet Filters in Plain English." November 10, 2004. <http://www.sans.org/rr/whitepapers/firewalls/798.php>.

15 – Howstuffworks. "How Routers Work." January 23, 2005. <http://computer.howstuffworks.com/router.htm>.

Last Updated: September 17th, 2019

Upcoming SANS Training

Click here to view a list of all SANS Courses

SANS San Francisco Fall 2019	San Francisco, CAUS	Sep 23, 2019 - Sep 28, 2019	Live Event
SANS Dallas Fall 2019	Dallas, TXUS	Sep 23, 2019 - Sep 28, 2019	Live Event
SANS London September 2019	London, GB	Sep 23, 2019 - Sep 28, 2019	Live Event
SANS Kuwait September 2019	Salmiya, KW	Sep 28, 2019 - Oct 03, 2019	Live Event
SANS Tokyo Autumn 2019	Tokyo, JP	Sep 30, 2019 - Oct 12, 2019	Live Event
SANS Cardiff September 2019	Cardiff, GB	Sep 30, 2019 - Oct 05, 2019	Live Event
SANS Northern VA Fall- Reston 2019	Reston, VAUS	Sep 30, 2019 - Oct 05, 2019	Live Event
SANS DFIR Europe Summit & Training 2019 - Prague Edition	Prague, CZ	Sep 30, 2019 - Oct 06, 2019	Live Event
Threat Hunting & Incident Response Summit & Training 2019	New Orleans, LAUS	Sep 30, 2019 - Oct 07, 2019	Live Event
SANS Riyadh October 2019	Riyadh, SA	Oct 05, 2019 - Oct 10, 2019	Live Event
SANS Baltimore Fall 2019	Baltimore, MDUS	Oct 07, 2019 - Oct 12, 2019	Live Event
SANS October Singapore 2019	Singapore, SG	Oct 07, 2019 - Oct 26, 2019	Live Event
SANS Lisbon October 2019	Lisbon, PT	Oct 07, 2019 - Oct 12, 2019	Live Event
SANS San Diego 2019	San Diego, CAUS	Oct 07, 2019 - Oct 12, 2019	Live Event
SIEM Summit & Training 2019	Chicago, ILUS	Oct 07, 2019 - Oct 14, 2019	Live Event
SANS Doha October 2019	Doha, QA	Oct 12, 2019 - Oct 17, 2019	Live Event
SANS Seattle Fall 2019	Seattle, WAUS	Oct 14, 2019 - Oct 19, 2019	Live Event
SANS SEC504 Madrid October 2019 (in Spanish)	Madrid, ES	Oct 14, 2019 - Oct 19, 2019	Live Event
SANS Denver 2019	Denver, COUS	Oct 14, 2019 - Oct 19, 2019	Live Event
SANS London October 2019	London, GB	Oct 14, 2019 - Oct 19, 2019	Live Event
SANS Cairo October 2019	Cairo, EG	Oct 19, 2019 - Oct 24, 2019	Live Event
SANS Santa Monica 2019	Santa Monica, CAUS	Oct 21, 2019 - Oct 26, 2019	Live Event
Purple Team Summit & Training 2019	Las Colinas, TXUS	Oct 21, 2019 - Oct 28, 2019	Live Event
SANS Training at Wild West Hackin Fest	Deadwood, SDUS	Oct 22, 2019 - Oct 23, 2019	Live Event
SANS Orlando 2019	Orlando, FLUS	Oct 28, 2019 - Nov 02, 2019	Live Event
SANS Houston 2019	Houston, TXUS	Oct 28, 2019 - Nov 02, 2019	Live Event
SANS Amsterdam October 2019	Amsterdam, NL	Oct 28, 2019 - Nov 02, 2019	Live Event
SANS DFIRCON 2019	Coral Gables, FLUS	Nov 04, 2019 - Nov 09, 2019	Live Event
Cloud & DevOps Security Summit & Training 2019	Denver, COUS	Nov 04, 2019 - Nov 11, 2019	Live Event
SANS Paris November 2019	Paris, FR	Nov 04, 2019 - Nov 09, 2019	Live Event
SANS Sydney 2019	Sydney, AU	Nov 04, 2019 - Nov 23, 2019	Live Event
SANS Mumbai 2019	Mumbai, IN	Nov 04, 2019 - Nov 09, 2019	Live Event
SANS Bahrain September 2019	OnlineBH	Sep 21, 2019 - Sep 26, 2019	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced