Global Information Assurance Certification Paper

Information Flow: Lessons Learned from the Old School

Christopher M. Spirito

March 5, 2001

Abstract:

Understanding how information flows is core to being able to protect that information in transport. When approaching the task of drafting an Organizational Security Policy, an Organization's Mission will set forth what information needs to flow where. Organizational INFOSEC guidance and Industry Regulations will dictate how information should flow. The Threat and Adversary model will outline how information should be protected. This information gathering process allows for a methodical approach to implementing technical and personnel countermeasures in order to protect and defend an Organization's information from compromise. Information Flow Analysis is critical to success. Neglecting this process will lead to compromise.

Introduction:

When I joined a DoD program a few years ago as a Security Architect, I had already spent a few years securing servers and managing security infrastructure devices, protecting organizations from the attackers of the world. So after listening to stories about crypto-gear falling off trucks during military operations, an older team member gave a briefing that included a process that analyzed Subjects and Objects and how they interact. At first I could only think that this old school mentality was just that, old school, complicating a process that was straight-forward, at least in my eyes. Why not just buy the hardware, install the OS, patches, applications, and more patches, then shut off everything that shouldn't be listening and add some ACLs and logging. Create a policy for backups and password management and let's move forward. This led to my first lesson from the Old School: approximative approaches to information security will lead to compromise: Information Flow Analysis is critical to success.

Subjects and Objects:

Understanding the relationship between Subjects and Objects is not difficult. Subjects are active entities such as processes and users. Objects are entities such as data structures [1]. Using Subjects and Objects, an Access Matrix can be constructed that illustrates which operations each Subject is allowed to apply to each Object. The following is a simple Access Matrix that defines relationships between a web-server specific Subjects and Objects:

		Objects			
		~/htdocs/*	**~/conf/***	**~/bin/***	**~/cgi-bin/***
Subjects	**process httpd**	read	read	read-execute	read-execute
	group webadmin	read-write-execute			
	user webdevel	read-write	none	none	read-write

Figure 1: Simple web-server Access Matrix

This Access Matrix allows the web server process, httpd, to only read from the Document Root, ~/htdocs, and the directory that holds the listener configuration files, ~/conf. The httpd process can execute items in the ~/bin and ~/cgi-bin directories. The webadmin group has access to everything in the web server installation tree. The user webdevel is only allowed to change files in the content directories. One could go the extra step and control configuration file changes and content updates, by forcing the webadmin group and webdevel user to modify and update files with a trusted tool that checks syntax and sets permissions on the files, not allowing for direct editing and writing of the file.

Least Privilege:

The Access Matrix for a system does a good job explicitly defining what interactions are allowed between Subjects and Objects. What is equally as important is what interactions are not allowed between Subjects and Objects. That is the principle of Least Privilege. The principle of Least Privilege requires that a user (or process running as that user) be given no more privilege than necessary to perform a job [2]. The System Administrator can approach this task by understanding the Objects and Subjects on their system, and how specific roles dictate specific capability. These concepts and methodologies now become the foundation of how we analyze the flow of information.

Organizational Security Policy:

To illustrate the importance of information flow we will briefly outline an Organizational Security Policy for an online initiative. Our mission is to provide to a community of clinician's information about drugs that are used to treat cardiac disorders. This information is stored in a database and will be delivered through a web-interface, dynamically generated based upon the clinician's request. The current

industry regulations require that each transaction be logged, keeping a record of which user-id requested what information at what time. Additional requirements exist that require frequent verification of the information through pre-scripted transactions to ensure the consistency and accuracy of the information. After briefly covering classes of attack, or areas of threat analysis, we will begin our information flow analysis. This information will assist us in completing the Threat and Adversary model and choosing appropriate technical and personnel countermeasures to put in place.

Classes of Attack:

When we begin our process of defining and analyzing the flow of information we must keep in mind the five classes of attack as defined in the IATFF Framework Document [3]:

- **Passive Attacks**: include traffic analysis, monitoring or unprotected communications, decrypting weakly encrypted traffic, and capturing authentication information.
- **Active Attacks**: include attempts to circumvent or break protection features, introduce malicious code, or steal or modify information.
- **Close-in Attack**: is where an unauthorized individual is in physical close proximity to networks, systems, or facilities for the purpose of modifying, gathering, or denying access to information.
- **Insider Attacks**: can be malicious or non-malicious. Malicious insiders have the intent to eavesdrop, steal or damage information, use information in a fraudulent manner, or denying access of other authorized users.
- **Distribution Attacks**: focus on the malicious modification of hardware or software at the factory or during distribution.

Information Flow:

There are two basic flows of information for this mission, viz: user retrieval of static content and user retrieval of dynamic content. The first case is straight-forward: a web-client makes a GET request to our web-server. The web-server returns static information contained in one or many files. Three threats associated with static information retrieval are:

1. Inadvertent release of static information to clinician's due to web-server misconfiguration or exploitation.
   Example: An attacker steals services by exploiting the web-server.
2. Denial of Service (DoS) to static information through known DoS attacks.
   Example: An attacker launches a distributed SYN-flooding attack against the web-server, thus denying service to paying members.
3. Malicious insertion, altering, and deleting of static information.
   Example: An attacker exploits the web-server to change static information about a drug, potentially causing a person to be given the wrong dosage.

The second case requires a little more due-diligence on our part. In the case of dynamically generated content, a web-client makes a GET or POST request, depending upon configuration, to our web-server. The web-server then executes a script or program, as the web-server user-id, to connect to a database and retrieve the requested information. This information is then processed and returned, often along with one or many files, to the web-client. Three threats associated with dynamic information retrieval are:

4. Poorly written scripts or programs that allow attackers to exploit the web-server and/or operating system.
   Example: A script or program that either doesn't handle errors correctly, causing the script or program to crash into an executing shell, or a script or program vulnerable to buffer overflows, allowing arbitrary commands to be issued on the system.
5. Neglecting context-checking to ensure the request does not contain information that could exploit the database.
   Example: A script or program that doesn't check the input fields, allowing an attacker to send hostile queries to a database.
6. Known vulnerable scripts and programs, distributed with the web-server, are left on the system and not disabled.
   Example: Vulnerability in the httpd nph-test-cgi script.

We have now identified some of the threats associated with the flow of information that is required to meet our mission requirements. The last set of threats are those that would allow an attacker an alternate path into our realm. Three threats that fall into this category are:

7. Exploitation of other services running on this system.
   Name Services (bind) daemon leads to root compromise of the system, leading to compromise of the web-server.
8. Denial of Service (DoS) to other services running on this system causing resources to be unavailable to business requests.
   Example: Distributed smurf or fraggle attacks against the server
9. Social Engineering.
   An attacker either directly or indirectly compromises a trusted web developer or system administrator, giving them knowledge and access to launch an attack on the systems and infrastructure.

We can assume our adversaries will range from script-kiddies out for fun to competitors who wish to cause harm to our business. Our most dreaded adversary is an attacker who changes drug information such that a person would receive a dangerous dose. In order to protect our information as it flows, appropriate countermeasures need to be put into place.

Technical Countermeasures (Network):

Focusing on the threats outlined above, we can implement Technical Countermeasures through the use of routers, firewalls, load-balancers, and leveraging wrappers on services running on the server. The following is a basic Access Matrix to protect the information as it flows across our network, from the edge to the web-server.

		Objects	
		TCP/UDP	**HTTP/HTTPS**
Subjects	**Internet Router**	allow tcp 80/443, udp 53 deny icmp, all	-
	Firewall	allow tcp 80/443 deny all	-
	Load Balancer or Proxy	allow 80/443 deny all	filter URL
	Web Server	allow tcp 22/80/443, icmp deny all	-

Figure 2: Simple Network-Device Access Matrix

Our Mission is to provide information via a web interface; therefore, all information requests will be via HTTP or HTTPS. The only other service that we are providing to our external customers is domain name services so end-users can find web addresses. The Internet Router will only allow packets to pass through with tcp.dst.port=80/443 and udp.dst.port=53. It will deny all external icmp requests and anything else that arrives. These are technical countermeasures to address the threats posed by smurf and fraggle attacks, to name two. We will assume our DNS server is up to date and non-exploitable to known attacks, although even if it is exploited, the attacker is contained outside of the firewall.

The firewall is a second layer of defense, receiving incoming packets and forwarding them into an enclave that contains the load balancer or proxy. At this point no external udp traffic will be forwarded and all traffic will be segmented from other operational environments. The firewall will also check for IP Spoofing and depending upon the vendor may assist in thwarting DoS attacks such as SYN-flooding. Lastly, the firewall will provide a method of logging connections for later analysis.

In addition to the usefulness of spreading traffic across multiple web-servers, load balancers and proxies are useful for URL filtering. This is important to support the protect->detect->react->respond/restore model. It may not always be possible to upgrade web-servers when exploits are announced. In order to protect and defend the computing environment, or web-server, it is important to have this filtering capability in place to mitigate the risk until the system is patched. The recent Unicode exploit on Microsoft's IIS (http://www.sans.org/y2k/unicode.htm) is an example of where URL filtering is necessary. In the time between when the exploit is announced and the patch is released and applied to the web-server, an important technical countermeasure would be to filter all URLs that start with: "http://www.domain.com/msadc".

This covers countermeasures that will address information as it flows from the boundary, through the infrastructure, to the web-servers. Although this is not covered, the services listening on the web-server, exclusive of httpd/httpsd, would have an additional Access Matrix defining what information could be processed by what processes. Thus far we have addressed threats #2, #7, #8. The use of load balancers or proxies also addresses threat #6, applied consistent with the previous paragraph about known vulnerabilities not being immediately patchable. This will also help to identify when attackers are probing around for vulnerabilities.

Technical Countermeasures (Host):

Our mission with regards to implementing technical countermeasures on the host is to ensure that the information processing applications are not exploitable and are carefully managed. This means that we have to make sure that our web-server is processing information consistent with our requirements, not allowing for additional rogue processing to take place. We also want to be able to do point-in-time checks of files for consistency, to make sure critical files have not been changed. Threat #3, the malicious insertion, altering, and deleting of static information can be addressed in at least two ways. First, the QA system that is part of our requirements can query for static pages and compare the results externally, allowing for an alert to be sent that a file has changed. This is also a good indicator that a new exploit may exist and steps should be taken to research this incident. Second, a host-based intrusion detection system can be deployed in order to check for file changes, including static web-content, system, and application files, allowing for an alert to be generated indicating an event to follow up on.

To address Threat #4 and #5 it takes a combination of technical countermeasures, Organizational INFOSEC guidance, and personnel countermeasures. It is essential that the scripts or programs that will pull the dynamic content out of the databases and return them to the end user are not exploitable due to known CGI exploit techniques. Organizational INFOSEC guidance should dictate programming methodologies for ensuring CGIs are not exploitable by known methods such as buffer-overflows. The following is an example of a poorly written CGI script, in C, and a better version that handles the buffer size [4]:

```
BAD CGI

#include
#include

static char query_string[1024];

char* read_POST() {
          int query_size;
          query_size=atoi(getenv("CONTENT_LENGTH"));
          fread(query_string,query_size,1,stdin);
          return query_string;
}
```

```
BETTER CGI

#include
#include

static char query_string[1024];

char* read_POST() {
          int query_size=atoi(getenv("CONTENT_LENGTH"));
          char* query_string = (char*) malloc(query_size);
          if (query_string != NULL)
                    fread(query_string,query_size,1,stdin);
          return query_string;
}
```

Additional precautions should be taken around scripts or programs that pass client-browser variables to system commands. If perl is used as a CGI scripting language, the web developers can use taint checks to make sure scripts do not pass environment variables to the shell. [5] Additional client-side checking using javascript or vbscript can also be added to prevent inadvertent buffer overflows by a user. The technical countermeasures that can be implemented can be derived from Figure 1 which dictates the objects with which the httpd process can interact. As stated in the Subjects and Objects and Least Privilege sections, the httpd process should only be given access to an essential set of applications. A system administrator should take the extra step and make sure that system commands that offer opportunities to gather intelligence, i.e. ipconfig, netstat, and offer opportunities for exploit, i.e. lynx for retrieval of files for later execution, have appropriate permissions set and/or renamed so that an attacker who is able to issue commands due to a server vulnerability, will not be able to achieve their goal of compromise as quickly or easily as expected, usually giving the intrusion detection systems enough time to discover the presence of an intruder. The personnel countermeasures that can be instantiated include independent QA review of scripts and programs before being put into production and a process that requires regular viewing of log files for signs of compromise.

Personnel Countermeasures:

With the implementation of personnel countermeasures we can approach the task of ensuring that the process of managing the web-server does not allow for inadvertent or malicious actions leading to information compromise. Referring to Figure 1 the Access Matrix indicates what actions the web-server process, webadmin administrator's group, and webdevel user are allowed to perform. The personnel countermeasure to address threat #1 is to only allow members of the webadmin group to update any configuration or binary files for the web-server. Additional care should be taken to ensure that all source and binary files are from trusted sources and trusted compilers. The web-server should also go through a QA process before being deployed, and after every notable change is made to its configuration to ensure that a vulnerability has not be introduced into the operational environment. The web-server configuration files and binaries should not writable by the web-server process, thus avoiding an exploit that does not need to elevate the privilege level to gain access to these critical files. Clearly there needs to be similar measures for management of the infrastructure devices between the border and the web-server.

The following list contains some additional personnel countermeasures that may help prevent both internal and external intruders from gaining access to resources not intended for their use [6]:

- Verifying the need for information access.
- Removing personal identifiers from access badges.
- Strict reporting of unusual or frequent contact with competitors and others with a marked interest in your product and operations.
- Monitoring of Internet Activity.
- Changing patterns when conducting sensitive operations.

The last threat to be addressed is #9, Social Engineering, one of the most difficult threats to detect and contain. The best personnel countermeasure to introduce is a thorough education program that covers methods of social engineering, how it can be prevented, and what suspicious activity should be reported and to whom. The Access Matrix that is developed for internal access should also take into

consideration much of the information presented in these sections, discussing the flow of information from the outside in. The flow of information within the enclave itself is critical, for many protocols pass information in cleartext across the network; information that can be used at a later date to compromise systems.

Conclusion:

The great thing about information flow analysis is that it provides the answers and justifications for why an infrastructure is built out the way it is. You can now say "I have this ruleset on this protection device to address this threat or this vulnerability." You can recommend technologies to your organization that meet the requirements dictated by the flow of information. You will know that your methodical analysis of the information, and how it travelled between objects, yielded an architecture that mitigates the threat of compromise or denial of service. A few years ago, I took the approximative approach and gave very fuzzy answers to these questions. Thanks to a lesson from the old school, my information is safer, and yours will be too.

References:

[1] "A Guide to Understanding Security Modeling in Trusted Systems (Aqua Book)." October 1992.
URL: http://www.radium.ncsc.mil/tpep/library/rainbow/NCSC-TG-010.pdf.

[2] "Role Based Access Control" (Ferraiolo & Kuhn, 1992), 15th National Computer Securuity Conference.
URL: http://hissa.ncsl.nist.gov/rbac/paper/rbac1.html.

[3] "IATFF Framework v3.0: Introduction" October 2000.
URL: http://www.iatff.net/login/framework_docs/version-3_0/pdffile.cfm?chapter=3ch01.

[4] "WWW Security FAQ: CGI Scripts v2.0.1" (Lincoln Stein, March 2000).
URL: http://www.w3.org/Security/Faq/wwwsf4.html.

[5] "WWW Security FAQ: Safe Scripting in Perl" (Lincoln Stein, March 2000).
URL: http://www.perl.com/pub/doc/FAQs/cgi/wwwsf5.html.

[6] "Countermeasures" (Dr. Ray, October 2000).
URL: http://polaris.umuc.edu/~lray/ifsm430/countermeasures.htm