Web Based Attacks

Attacks upon information security infrastructures have continued to evolve steadily overtime; legacy network based attacks have largely been replaced by more sophisticated web application based attacks. This paper will introduce and address web based attacks from attack to detection. Information security professionals new to application layer attacks will be in a better position to understand the underlying application attack vectors and methods of mitigation after reading this paper.

Web Based Attacks

GCIA Gold Certification

Author: Justin Crist, jcrist@secureworks.com

Adviser: Jim Purcell

Abstract

Attacks upon information security infrastructures have continued to evolve steadily overtime; legacy network based attacks have largely been replaced by more sophisticated web application based attacks. This paper will introduce and address web based attacks from attack to detection. Information security professionals new to application layer attacks will be in a better position to understand the underlying application attack vectors and methods of mitigation after reading this paper.

Table of Contents

Introduction:

Before discussing web application security or attacks it is vitally important to understand the evolution of web applications, their increasing complexity and the paramount importance that they play in over one billion peoples' lives today (Internet World Stats).

The advent of first generation web applications was severely limited in their ability to provide any more information than a brochure you might receive in the mail. Static HTML was provided as a tool to display pictures and inert information. Consequently, as the internet and web access became more and more ubiquitous so too did the needs of those users who were accessing web applications. As a result web applications evolved to provide user conveniences such as searching, posting, and uploading.

CGI, Common Gateway Interface protocol was the first leap forward in this progression. CGI provided a means for users to interact with web pages by submitting data into forms. Upon submission back end CGI scripts would process this data presented and represent HTML back to the end user. CGI through the interaction with end users effectively became one of the first web application attack vectors known.

As mentioned earlier web application development did not stop with CGI scripts, instead newer more evolved frameworks manifested. PHP, ASP.NET, J2EE, AJAX, Ruby on Rails, and others emerged to incorporate more interactive features which allow users more flexibility and power when managing data and workflow within web applications.

Securing web applications has become incredibly important as the information processed by web applications has become critical to corporations, customers, organizations, and countries. Web applications manage a wide array of information including financial data, medical records, social security numbers, intellectual property and national security data. Web applications must handle this information securely while maintaining efficiency and availability.

What is a web based attack?

Web based attacks are considered by security experts to be the greatest and oftentimes the least understood of all risks related to confidentiality, availability, and integrity.(cite) The purpose of a web based attack is significantly different then other attacks; in most traditional penetration testing exercises a network or host is the target of attack. Web based attacks focus on an application itself and functions on layer 7 of the OSI. John Pescatore of the Gartner group claims that nearly 70% of all attacks occur at the application layer (Desmond, 2004).

Application vulnerabilities could provide the means for malicious end users to breach a system's protection mechanisms typically to take advantage or gain access to private information or system resources. Information gathered can include social security numbers, dates of birth, and maiden names, which are all often used in identity theft. Another popular target for attackers is credit card data which left unprotected and unencrypted can be used to cause significant damage to organizations most valued assets, their customers.

So what makes up an application attack? By definition, all web application attacks are comprised of at least one normal request or a modified request aimed at taking advantage of poor parameter checking or instruction spoofing. There are six fundamental categories of application attacks.

Spoofing:

Spoofing is the act of mimicking another user or process to perform a task or retrieve information that would normally not be allowed. An attacker could use a crafted HTTP request containing the session id information from another user and retrieve the targeted users account information.

Repudiation:

In order to tie specific actions of a single user, applications must have reasonable repudiation controls such as web access, authentication, and database transaction logs. Without corroborating logs, online web application users could easily claim that they did not transfer equities from one acct to an external acct of another. Otherwise without proof otherwise all online brokerages would be required to reimburse the client for lost funds. Aggregating and correlating logs from multiple sources (web application, middleware, and database) can prevent repudiation attacks.

Information Disclosure:

Information disclosure is one of the biggest threats to large organizations who maintain private information about their customer base. When attackers are capable of revealing private information about a user or users of a web site, consumer confidence in that organization can take drastic hits; causing loss in sales, stock price, and overall marketability. To prevent this, applications must require adequate controls which will prevent user ID and session manipulation.

Denial of Service:

Denial of service attacks are likely the most well known of all application attacks, often generated by malicious users, competitors or script kiddies. Motivations for this type of an attack range from personal to political reasons in hopes of stifling an organization’s ability to field online business. Famous examples include attacks upon SCO a couple of years ago by individuals upset about lawsuits aimed at LINUX.

Elevation of Privileges:

Authorization controls which are both reliable and staunch are requisite for any system or application which guards sensitive information. Escalation of privileges requires a malicious user to either already possess or gain through unlawful methods authorization privileges of a regular user. Once the malicious user is logged into the victim system an attempt will be made by exploitation of an application through poor parameter checking or instruction spoofing.

Who is at risk from Web Based Attacks

All organizations which maintain a web presence are at risk of being attacked. However, the level of risk is different for each organization. A couple of factors that play into consideration when determining the threat level are intellectual property or personally identifiable information stored by the organization.

Intellectual property:

Intellectual property is a product of the intellect that has commercial value, including copyrighted property such as literary or artistic works, and ideational property, such as patents, appellations of origin, business methods, and industrial processes. Companies where success and reputation are built upon patents, research, and development are especially at risk. Great examples include pharmaceutical companies, chip fabricators and universities.

Personally Identifiable Information:

Nearly all organizations which interface with customers hold to some degree information which is considered to be sensitive. For attackers to perpetrate financial crimes or identity theft they need credit card numbers, phone numbers, addresses, health related information, bank account numbers (Krebs, 2006). As such this information has become a commodity in underground IRC chat rooms.

Three Aspects of a Web Based Attack

Vulnerability Prevention:

The first step in a comprehensive application security framework starts with developers. Software architecture just like building physical structures requires sound planning and oversight, with an adherence to fundamental software development lifecycle methodologies. Many software developers carry the skills to properly proofread and locate vulnerabilities. However as an insurance policy of sorts exploitation detection tools are required to provide a standard level of error checking.

Attack:

It for first necessary for individuals interested in conducting an application based attack it is first necessary to understand and identify a target system. This first stage in an attack is commonly referred to as reconnaissance; reconnaissance can be performed using a variety of tools such as port scanners and vulnerability scanners. These scanners are often flexible enough to look for specific listening ports associated with suspected vulnerable applications. Application version detection can be performed by banner grabbing. Banner grabbing is the process of connecting to a host on a specific TCP/UDP port and listening to what the host replies with. Once a connection is established applications will commonly identify or provide a version or build information relevant to the application that host is using. By grabbing banners attackers are quickly able to cross reference application versions, patch levels, and build information to online references which list vulnerable applications. In addition to online resources which list vulnerable applications many freely available tools contain self contained and dynamically updating databases which perform application mapping to current vulnerabilities. A closer in depth look at several of these will follow in a later section.

Attack Detection & Prevention:

Intrusion Detection Systems (IDS) and Intrusion Prevention systems (IPS) are also used as detective and reactive devices for both network and application targeted attacks. They are commonly deployed within the demilitarized zones (DMZs) of corporate networks and either passively filter or actively block attacks targeted at application layer services. IDSs and IPSs both work at the network layer by listening to network traffic destined to protected systems for attacks against vulnerable services, data manipulation attacks on applications, privilege escalation on hosts, multiple failed unauthorized logins, and even access to sensitive data. Upon successful detection; IDS alerts are usually sent to a central console where action by an analyst can be taken. Intrusion Prevention Systems upon detection of an attack, can drop the packets, send TCP resets to the offending IP or shun further connections by malicious attackers for a variable period of time. Some Intrusion Detection Systems have the ability to thwart offending attackers by communicating directly with a firewall or router to block the source IP address.

Reverse web proxies are one of many tools that can be used to not only detect, but protect an application server from unauthorized, malicious, or inappropriate calls, posts, or queries to online web applications, services and databases. A reverse web proxy works by intercepting traffic destined to a protected web application or service and then applies filters to detect malicious commands, bad syntax, inappropriate content, and the like. By standing in front of the web applications, this allows the web application to focus on legitimate requests.

Many firewalls sold in the market place today are application aware and as such understand many of the ubiquitous protocols and commands. In understanding these protocols and applications they are able to ascertain whether or not traffic destined for an application or network based service is malicious or not. Common applications include telnet, SSH, HTTP, FTP, SMTP and SIP. When tuned properly these firewalls are capable of thwarting many common attacks waged against vulnerable applications or protocols. Upon inspection of malicious payloads or commands sent to an application, these advanced firewalls will usually offer the ability to drop malicious packets, enact temporary or permanent filters against all traffic destined from the offending IP, or send an alert to security personnel so that further action can be taken.

Prevention/Detection Methods

Many methodologies exist, and there is never one right solution or architecture for all environments. Previously discussed, there are more passive methods which include Intrusion Detection Systems (IDS). IDS systems simply alert on seemingly offensive traffic which is destined towards a protected asset or application. On the opposite spectrum there are more preventative methods which include application-aware firewalls, reverse proxies, and Intrusion Detection Prevention (IDP) which not only actively monitor for attacks but they attempt to block or change the environment such that further attacks are not successful in reaching the protected application or system.

For environments which contain known off the shelf applications which are using common protocols such as FTP, HTTP, HTTPS, SMTP, and others IDP can be very effective in protecting assets in a reactive fashion. For those larger environments which contain home grown applications, oddly configured application and service ports, or systems which do not follow RFC standards when communicating, IDS may be a more intelligent solution.

The reason is simple; IDP solutions typically block malicious traffic, or traffic which does not adhere to RFC standards. RFC standards or Requests for Comments are rules commissioned by the IETF which dictate proper communication methods and practices for nearly all well known protocols in use on the internet today. Many third party or home grown applications are built with one purpose in mind, to work. The adherence of a particular application or service to an RFC standard is very rarely a priority for application developers.

As such, applications, especially legacy applications are often put together in a hasty fashion with little regard with how it may or may not be affected by a security device such as IDP. It should be understood that IDP has only been around for a couple of years and is still a maturing technology.

Additionally log analysis on critical files which are used by both applications and their underlying systems can provide good indications as to when an intrusion or unsuccessful penetration has occurred. There are a variety of both open source and proprietary tools which offer the ability to monitor key configuration files (integrity checker), and log files (log correlation and monitoring tools). We will discuss examples of each of these in the next section.

Log Monitoring and Correlation Tools:

Gartner has identified SIEMs or Security Information and Event Managers as a means to provide "real-time event management and historical analysis of security data from a wide set of heterogeneous sources" (Gartner, 2005).

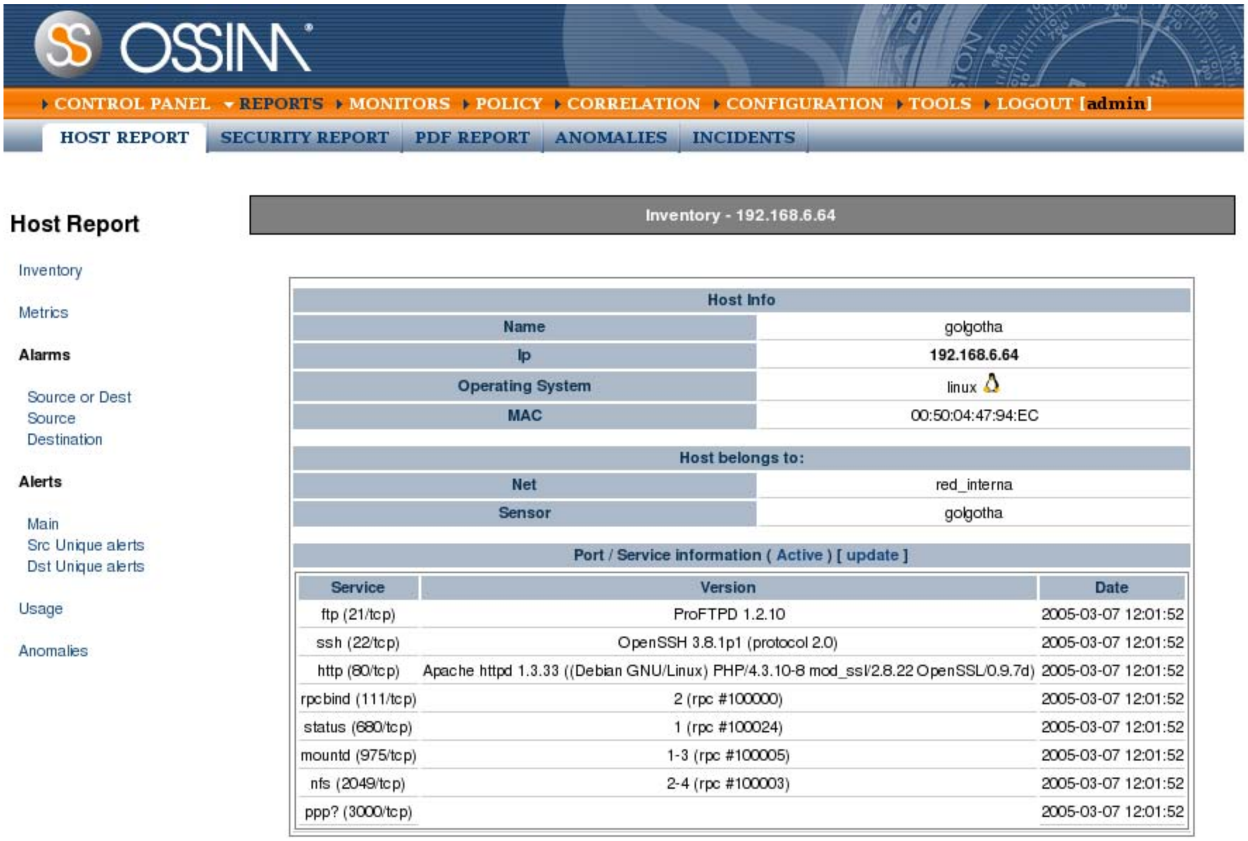
In larger organizations which often feature a multitude of servers spread across different geographies, log analysis and review becomes increasingly difficult and labor some. "This stream of events and alerts—Gartner estimates that the systems in companies with more than 1000 users generate over 200 security "events" per second—is enough to overwhelm any IT department", notes Jason Halloway of ExaProtect(Halloway, ExaProtect).

In this example, if an administrator or security minded engineer wishes to proactively review the logs of his or her managed systems a number of steps must ensue.

1. It would be necessary to remotely login to these systems one at a time.
2. Filter log files for interesting events.
3. Match these ‘interesting’ events with other similar events on other managed systems.

This process was identified as far too time consuming, tedious, and prone to errors or mistakes. With this problem came the advent of log monitoring and correlation tools which serve to filter and match a log event on one device or application to a complimentary or similar event on another system. Being privy to the logs or events which are occurring on different systems can lead to more intelligent decisions about the source of an attack, a misconfiguration of an application or system, or even an internal user who wishes to cause harm.

SIEM solutions are prevalent and there are a number of both proprietary and open source alternatives available. ArcSight, netForensics, Sentinel, and Intellitactics are but a few proprietary solutions that are available (Gartner, 2006). Open source solutions are also attractive for smaller businesses that may not have the budget for proprietary solutions which oftentimes offer better scale and more robust support. OSSIM, or the Open Source Security Information Management tool is a popular choice amongst the open source community. It offers many of the features of proprietary solutions without the high cost and licensing fees. A few screen shots have been added below.



Host Report

Inventory

Metrics

Alarms

Source or Dest
Source
Destination

Alerts

Main
Src Unique alerts
Dst Unique alerts

Usage

Anomalies

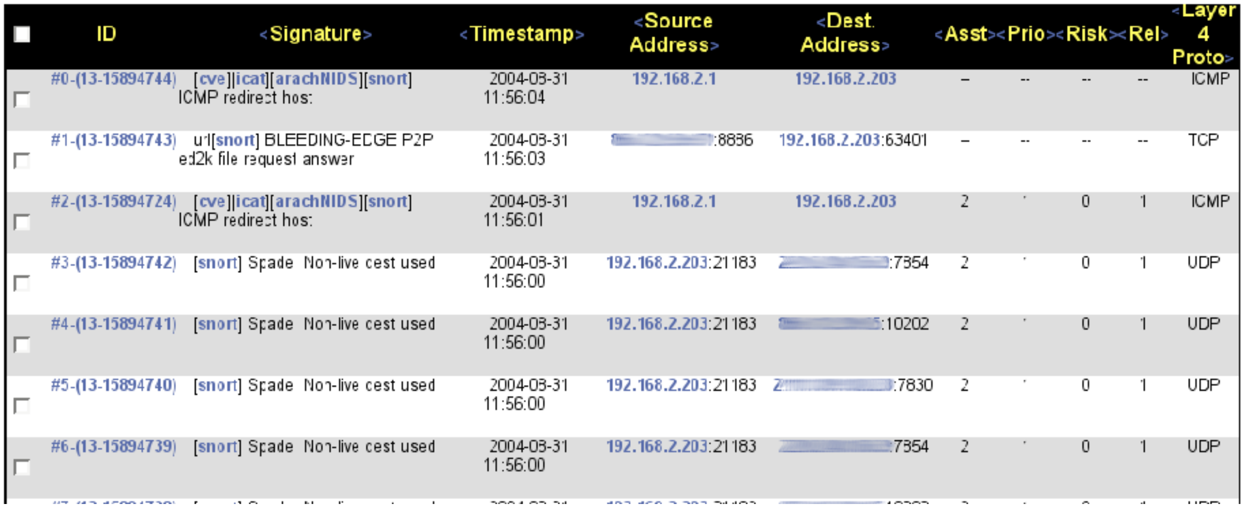
Inventory - 192.168.6.64

Host Info	
Name	golgotha
Ip	**192.168.6.64**
Operating System	linux
MAC	00:50:04:47:94:EC
Host belongs to:	
Net	red_interna
Sensor	golgotha

Port / Service information (Active) [update]

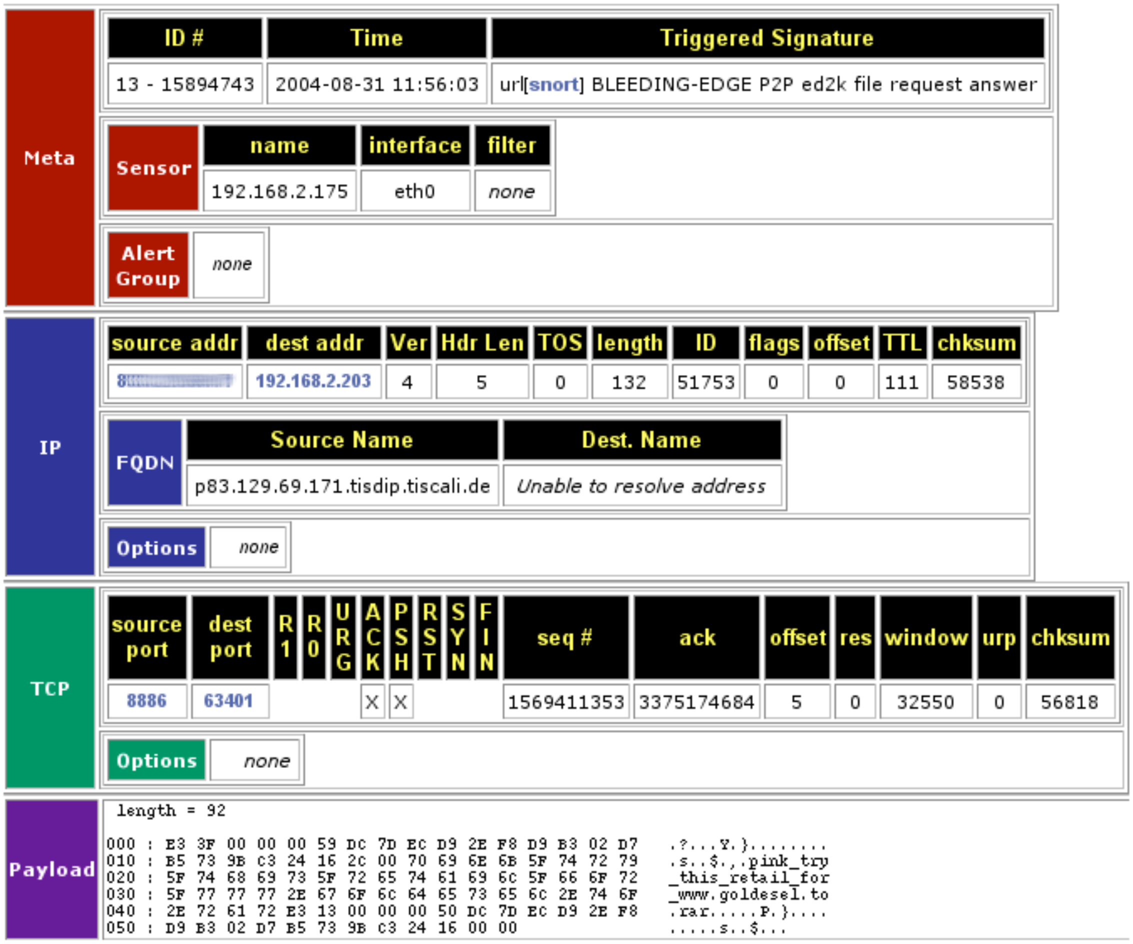
Service	Version	Date
ftp (21/tcp)	ProFTPD 1.2.10	2005-03-07 12:01:52
ssh (22/tcp)	OpenSSH 3.8.1p1 (protocol 2.0)	2005-03-07 12:01:52
http (80/tcp)	Apache httpd 1.3.33 ((Debian GNU/Linux) PHP/4.3.10-8 mod_ssl/2.8.22 OpenSSL/0.9.7d)	2005-03-07 12:01:52
rpcbind (111/tcp)	2 (rpc #100000)	2005-03-07 12:01:52
status (680/tcp)	1 (rpc #100024)	2005-03-07 12:01:52
mountd (975/tcp)	1-3 (rpc #100005)	2005-03-07 12:01:52
nfs (2049/tcp)	2-4 (rpc #100003)	2005-03-07 12:01:52
ppp? (3000/tcp)		2005-03-07 12:01:52

Here we can see signature matches by source address, destination, etc. These can be sorted by the various columns and can help security analysts determine common denominators in an attack.



□	ID	<Signature>	<Timestamp>	<Source Address>	<Dest. Address>	<Asst>	<Prio>	<Risk>	<Rel>	<Layer 4 Proto>
□	#0-(13-15894744)	[cve][icat][arachNIDS][snort] ICMP redirect host	2004-03-31 11:56:04	192.168.2.1	192.168.2.203	–	--	--	--	ICMP
□	#1-(13-15894743)	url[snort] BLEEDING-EDGE P2P ed2k file request answer	2004-03-31 11:56:03	[redacted]:8836	192.168.2.203:63401	–	--	--	--	TCP
□	#2-(13-15894724)	[cve][icat][arachNIDS][snort] ICMP redirect host	2004-03-31 11:56:01	192.168.2.1	192.168.2.203	2	'	0	1	ICMP
□	#3-(13-15894742)	[snort] Spade: Non-live dest used	2004-03-31 11:56:00	192.168.2.203:21183	[redacted]:7354	2	'	0	1	UDP
□	#4-(13-15894741)	[snort] Spade: Non-live dest used	2004-03-31 11:56:00	192.168.2.203:21183	[redacted]:10202	2	'	0	1	UDP
□	#5-(13-15894740)	[snort] Spade: Non-live dest used	2004-03-31 11:56:00	192.168.2.203:21183	[redacted]:7830	2	'	0	1	UDP
□	#6-(13-15894739)	[snort] Spade: Non-live dest used	2004-03-31 11:56:00	192.168.2.203:21183	[redacted]:7354	2	'	0	1	UDP

This screenshot shows the actual packet broken down into the different layers of the TCP/IP stack. This will be helpful in analyzing IDS signatures of valid attacks and even misconfigurations in applications and systems.



Meta

ID #	Time	Triggered Signature
13 - 15894743	2004-08-31 11:56:03	url[snort] BLEEDING-EDGE P2P ed2k file request answer

Sensor	name	interface	filter
	192.168.2.175	eth0	*none*

Alert Group	*none*

IP

source addr	dest addr	Ver	Hdr Len	TOS	length	ID	flags	offset	TTL	chksum
[redacted]	192.168.2.203	4	5	0	132	51753	0	0	111	58538

FQDN	Source Name	Dest. Name
	p83.129.69.171.tisdip.tiscali.de	*Unable to resolve address*

Options	*none*

TCP

source port	dest port	R 1	R 0	URG	ACK	PSH	RST	SYN	FIN	seq #	ack	offset	res	window	urp	chksum
8886	63401				X	X				1569411353	3375174684	5	0	32550	0	56818

Options	*none*

Payload

```
 length = 92

000 : E3 3F 00 00 00 59 DC 7D EC D9 2E F8 D9 B3 02 D7   .?...Y.}........
010 : B5 73 9B C3 24 16 2C 00 70 69 6E 6B 5F 74 72 79   .s..$.,.pink_try
020 : 5F 74 68 69 73 5F 72 65 74 61 69 6C 5F 66 6F 72   _this_retail_for
030 : 5F 77 77 77 2E 67 6F 6C 64 65 73 65 6C 2E 74 6F   _www.goldesel.to
040 : 2E 72 61 72 E3 13 00 00 00 50 DC 7D EC D9 2E F8   .rar.....P.}....
050 : D9 B3 02 D7 B5 73 9B C3 24 16 00 00               .....s..$...
```

All screenshots were taken from www.ossim.net.

OSSIM: For detecting attacks

The larger a given organizations' infrastructure the more difficult it becomes to be proactive about detecting attacks before they can burgeon into wide scale problems. The reasons are numerous but can be narrowed to the following:

Systems are often distributed across disparate networks in different parts of a business unit, country, and even continent.

Diverse organizations, teams, and business units with different attitudes manage these disparate networks. As a result patching, build procedures, network security, and general security best practices can take a back seat to other revenue generating activities.

Oftentimes different configuration standards and policies may be the result of regulatory requirements, government restrictions, and government supported export restrictions. It quickly becomes obvious that Security Analysts are faced with a staggering task. In order to perform the job successfully security analysts are required to understand a network topology that may span a multitude of countries. Achieving this thorough understanding of the lay of the land is often much easier said than done.

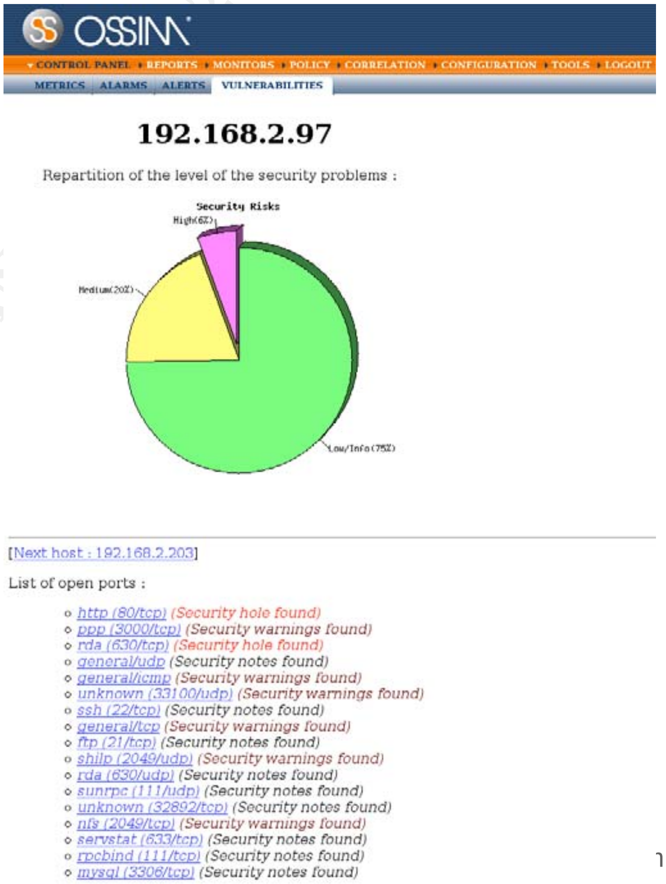
To assist, SIM packages coupled with network reconnaissance tools are often used to map out network topologies. OSSIM is an open source Security Information Manager that includes Nessus, a very popular and freely available vulnerability scanner and NMAP, often considered to be one of the best available network mapping tools. Both Nessus and NMAP possess the ability to map network spaces. Network probing and the different methods of network discovery are beyond the scope of this paper. To comprehend the forthcoming example it is only necessary to understand that Nmap allows a security analyst to scan networks and visually map these networks. These pictures or visualizations allow security analysts to understand the inner workings of their mapped networks. Without these maps, learning curves are significantly steeper, and often breed unnecessary confusion amongst security analysts responsible for the network,

network users, and network administrators. A layman can think of a network map as a street map. Street maps encourage common understanding by those that use the roads and those that provide directions.

For this particular example we will be using Nmap and Nessus. First, Nmap is initiated to probe IP address space which is owned by the parent organization of the security analyst. The results of this scan will yield all listening devices with a TCP/IP stack that are responsive to a variety of ICMP, TCP, and UDP sweeps.

Following the successful run of Nmap, A Nessus scan is initiated from within OSSIM against previously discovered network hosts for vulnerabilities. After scanning is completed a list of known vulnerabilities of the scanned hosts are stored within the OSSIM database.

Nessus sample output for one host has been displayed below.



192.168.2.97

Repartition of the level of the security problems :

[Next host : 192.168.2.203]

List of open ports :

- http (80/tcp) (Security hole found)
- ppp (3000/tcp) (Security warnings found)
- rda (630/tcp) (Security hole found)
- general/udp (Security notes found)
- general/icmp (Security warnings found)
- unknown (33100/udp) (Security warnings found)
- ssh (22/tcp) (Security notes found)
- general/tcp (Security warnings found)
- ftp (21/tcp) (Security notes found)
- shilp (2049/udp) (Security warnings found)
- rda (630/udp) (Security notes found)
- sunrpc (111/udp) (Security notes found)
- unknown (32892/tcp) (Security notes found)
- nfs (2049/tcp) (Security warnings found)
- servstat (633/tcp) (Security notes found)
- rpcbind (111/tcp) (Security notes found)
- mysql (3306/tcp) (Security notes found)

Because there are nearly always going to be emerging threats and unpatched vulnerabilities is it necessary to create a risk rating. OSSIM uses the following methodology to quantify risk. This is important to understand for the security analyst. These quantifications of risk allow an analyst to prioritize his or her attention towards the most burning issues.

Risk = (Asset * Priority * Reliability)/10

Where:

Asset (0-5)

Priority (0-5)

Reliability (0-5)

The computed risk rating will always be between 0 and 10.

Asset Value

In determining asset values, the asset owner or data owner is generally responsible for classifying this for the security analyst. This step should not be overlooked as it enables security analysts to have an understanding of criticality amongst the various network hosts. Based upon this feedback an asset can be tagged with a value from 0-5. Customer data, intellectual property and the like might be tagged with a 4 or 5. While the system that stores the lunch menu for the cafeteria may receive a lower score, 0. These numerical values help an organization prioritize their security efforts, DR, and other contingency plans.

Priority/Threat

Priority or threat refers to the significance of a particular attack on a managed asset. In other words, what might be the disruption to my environment in the event that a particular vulnerability is being exposed? OSSIM is able to make decisions relative to the environment that is under attack regarding priority.

A common scenario which OSSIM is able to help is described in the following scenario.

A Microsoft Windows targeted virus code red has infiltrated the corporate network from the internet. Through the use of snort, a popular open source Intrusion Detection System, snort is able to detect that code red, an IIS targeted worm is running rampant on the DMZ. Due to the sheer number of hosts that are found within the DMZ a security analyst can quickly find herself in a tenuous position of not knowing which hosts to protect first.

Though no security analyst should condone the spreading of worms in their network, it is safe to say that Windows IIS vulnerabilities will have little to no affect on a UNIX or Linux based server. At the same time as this code red attack is taking place against a UNIX server farm it also has the ability to spread to the corporate intranet subnet which does contain unpatched Windows Servers running IIS. Applying higher priorities for this behavior in a windows environment allows a security analyst to respond more appropriately often decreasing the time for remediation.

To properly prioritize OSSIM provides a method to prioritize threat levels. We will dive deeper into these directives in a subsequent section.

Reliability

Reliability speaks specifically to the likelihood that an alert which is logged by OSSIM is worthy of an alarm. An alert is simply any event that is logged from one of the many devices which logs to OSSIM. This could be a log received from a router, firewall, or host on the network. Reliability starts to take into account patterns and characteristics in the events that are being logged. For example, if a windows workstation connects to on average of 20-30 hosts on TCP 135 during a given hour this might be considered normal based upon the duties of this windows host. However, if this same system begins to connect to 400-600 hosts in a half hour it is reasonable to suspect the presence of a misconfiguration or even perhaps the presence of a compromised host.

Reliability of an alarm speaks to the likelihood that the alarm is accurately depicting a certain scenario which more times than not need to be acted upon.

Alarms within OSSIM can be triggered after a certain number of like events have been received. Coupled with their asset values, and priorities, a risk rating can be computed.

Power of Correlation

As mentioned before directives are used within OSSIM. Directives are in essence rules or formulas which are used to calculate the potential classification or meaning of a series of received alerts.

For those familiar with CISCO based ACLs, directives have a somewhat similar nomenclature which use numbers as identifiers.

- Generic ossim: 1-2999

- Attack correlation: 3000-5999
- Virus & Worms: 6000-8999
- Web attack correlation: 9000-11999
- DoS: 12000-14999
- Portscan/scan: 15000-17999
- Behaviour anomalies: 18000-20999
- Network abuse & error: 21000-23999
- Trojans: 24000-26999
- Miscelaneous: 27000-34999

Directives all start with an initial rule that attempts to match the directive and begin the event correlation. For example, given the example above where a windows host connects to 400 windows systems on port 135.

We will illustrate this with a detector rule.

```
<rule type = “detector” name=”Potentially Suspicious”
reliability = “1” occurrence=”500” from=”ANY” to=”ANY”
port_to=”135,137,139,445” plugin id=”50051” plugin_sid=”ANY”
time_out=360>
```

For this particular rule we match against anything going to standard windows port which exceeds 500 occurrences or connections in a one hour time span.

Detector rules are rules that are received from agents which are recording to OSSIM. Examples may include apache, snort, fW-1, etc. The power of OSSIM and correlation is the ability to receive information from disparate sources to arrive at a deeper conclusion or understanding of what is occurring in a network.

Below we have broken down the major categories within a directive.

Name field simply represents the proper name of the directive.

Reliability is based upon the likelihood that this event is noteworthy. Again the scale of 0 – 5 is used.

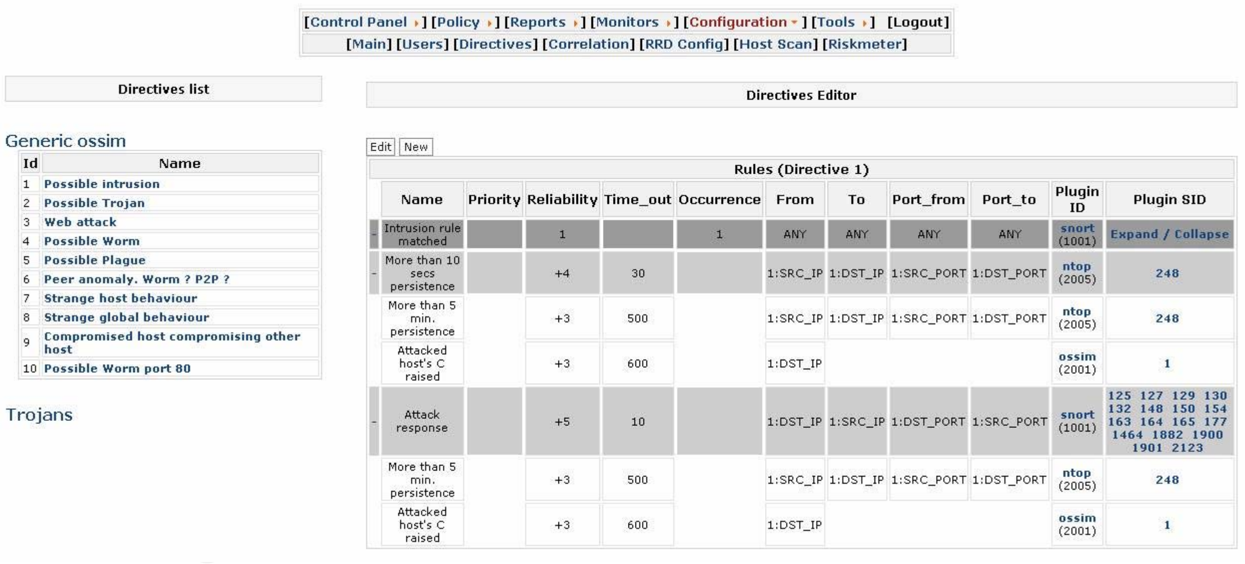
Occurrence indicates how many times a particular event will have to occur before the directive is matched.

From indicates the source of the event.
To indicates the destination of the event.

Ports_to indicate the source port of the event.
Ports_from is not listed here but can be defined for source port matching.

Events which occur by themselves oftentimes are not particularly noteworthy; however, when multiple sources are recording similar behaviors which differ from the norm behaviors, OSSIM is able to alert security analysts of this potentially interesting activity. The power that correlation can offer a security analyst should not be underestimated.

The following screen shot shows the structure visually of a directive that is built within OSSIM.


[Control Panel ▸] [Policy ▸] [Reports ▸] [Monitors ▸] [Configuration ▾] [Tools ▸] [Logout]
[Main] [Users] [Directives] [Correlation] [RRD Config] [Host Scan] [Riskmeter]
Edit
New
Expand / Collapse

Directives list

Generic ossim

Id	Name
1	Possible intrusion
2	Possible Trojan
3	Web attack
4	Possible Worm
5	Possible Plague
6	Peer anomaly. Worm ? P2P ?
7	Strange host behaviour
8	Strange global behaviour
9	Compromised host compromising other host
10	Possible Worm port 80

Trojans

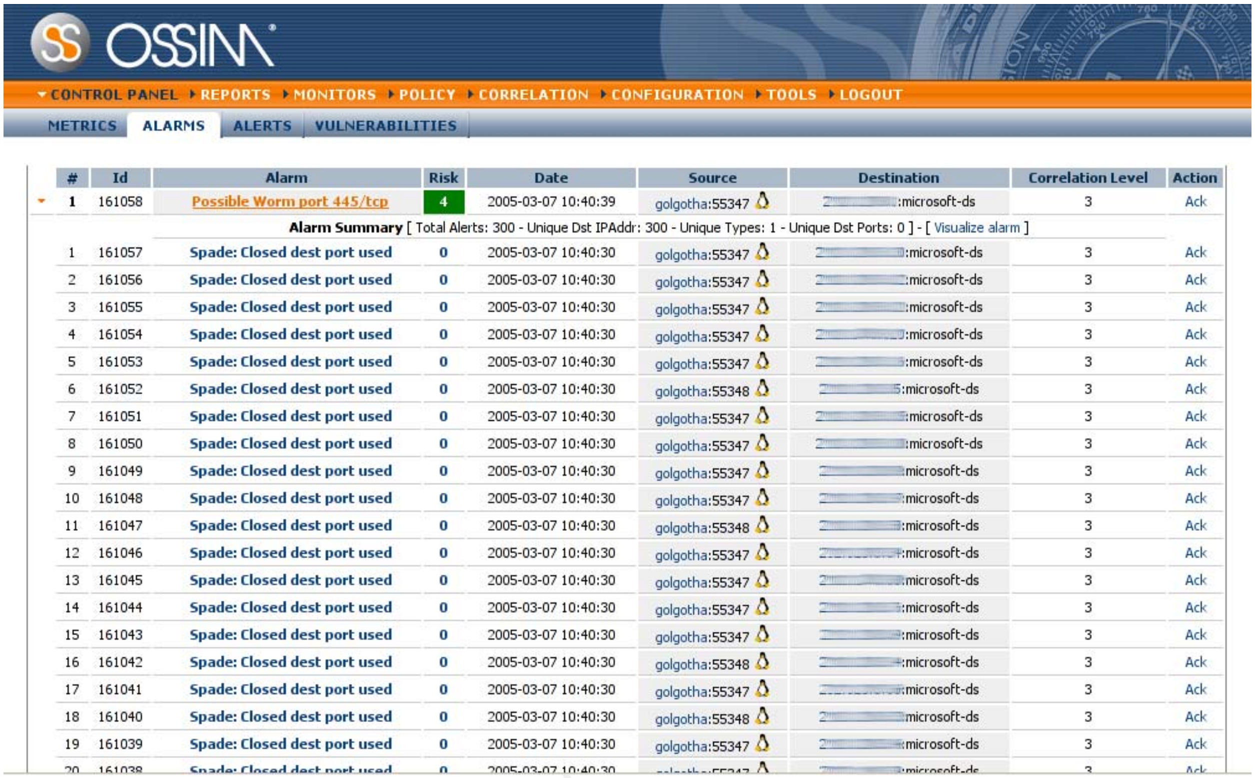
Directives Editor

Rules (Directive 1)

	Name	Priority	Reliability	Time_out	Occurrence	From	To	Port_from	Port_to	Plugin ID	Plugin SID
-	Intrusion rule matched		1		1	ANY	ANY	ANY	ANY	snort (1001)	Expand / Collapse
-	More than 10 secs persistence		+4	30		1:SRC_IP	1:DST_IP	1:SRC_PORT	1:DST_PORT	ntop (2005)	248
	More than 5 min. persistence		+3	500		1:SRC_IP	1:DST_IP	1:SRC_PORT	1:DST_PORT	ntop (2005)	248
	Attacked host's C raised		+3	600		1:DST_IP				ossim (2001)	1
-	Attack response		+5	10		1:DST_IP	1:SRC_IP	1:DST_PORT	1:SRC_PORT	snort (1001)	125 127 129 130 132 148 150 154 163 164 165 177 1464 1882 1900 1901 2123
	More than 5 min. persistence		+3	500		1:SRC_IP	1:DST_IP	1:SRC_PORT	1:DST_PORT	ntop (2005)	248
	Attacked host's C raised		+3	600		1:DST_IP				ossim (2001)	1

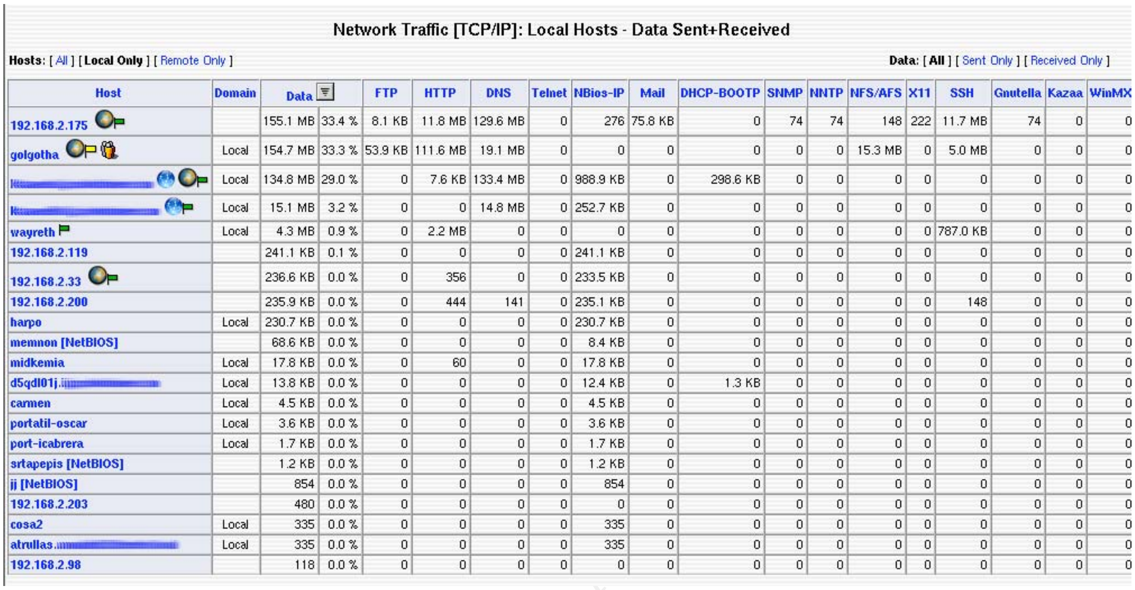
Justin Crist

Below you will find a screen shot taken from OSSIM.net which indicates the likelihood of a windows worm. Notice the correlation level, risk rating, and alarm.



#	Id	Alarm	Risk	Date	Source	Destination	Correlation Level	Action
1	161058	Possible Worm port 445/tcp	4	2005-03-07 10:40:39	golgotha:55347	[redacted]:microsoft-ds	3	Ack
		Alarm Summary [Total Alerts: 300 - Unique Dst IPAddr: 300 - Unique Types: 1 - Unique Dst Ports: 0] - [Visualize alarm]						
1	161057	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55347	[redacted]:microsoft-ds	3	Ack
2	161056	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55347	[redacted]:microsoft-ds	3	Ack
3	161055	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55347	[redacted]:microsoft-ds	3	Ack
4	161054	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55347	[redacted]:microsoft-ds	3	Ack
5	161053	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55347	[redacted]:microsoft-ds	3	Ack
6	161052	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55348	[redacted]:microsoft-ds	3	Ack
7	161051	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55347	[redacted]:microsoft-ds	3	Ack
8	161050	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55347	[redacted]:microsoft-ds	3	Ack
9	161049	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55347	[redacted]microsoft-ds	3	Ack
10	161048	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55347	[redacted]microsoft-ds	3	Ack
11	161047	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55348	[redacted]:microsoft-ds	3	Ack
12	161046	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55347	[redacted]:microsoft-ds	3	Ack
13	161045	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55347	[redacted]microsoft-ds	3	Ack
14	161044	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55347	[redacted]:microsoft-ds	3	Ack
15	161043	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55347	[redacted]:microsoft-ds	3	Ack
16	161042	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55348	[redacted]:microsoft-ds	3	Ack
17	161041	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55347	[redacted]:microsoft-ds	3	Ack
18	161040	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55348	[redacted]microsoft-ds	3	Ack
19	161039	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55347	[redacted]microsoft-ds	3	Ack
20	161038	Spade: Closed dest port used	0	2005-03-07 10:40:30	golgotha:55347	[redacted]microsoft-ds	3	Ack

Similarly the following screens can be used to corroborate any alarms which may be sent. Analysts have the ability to determine through this snapshot which hosts are responsible for sending and receiving the majority of the network traffic. Other screens indicate the timing of network usage. This information by itself may not be particularly telling, however, with the use of the previous screen, certain hypothesis begin to take hold.



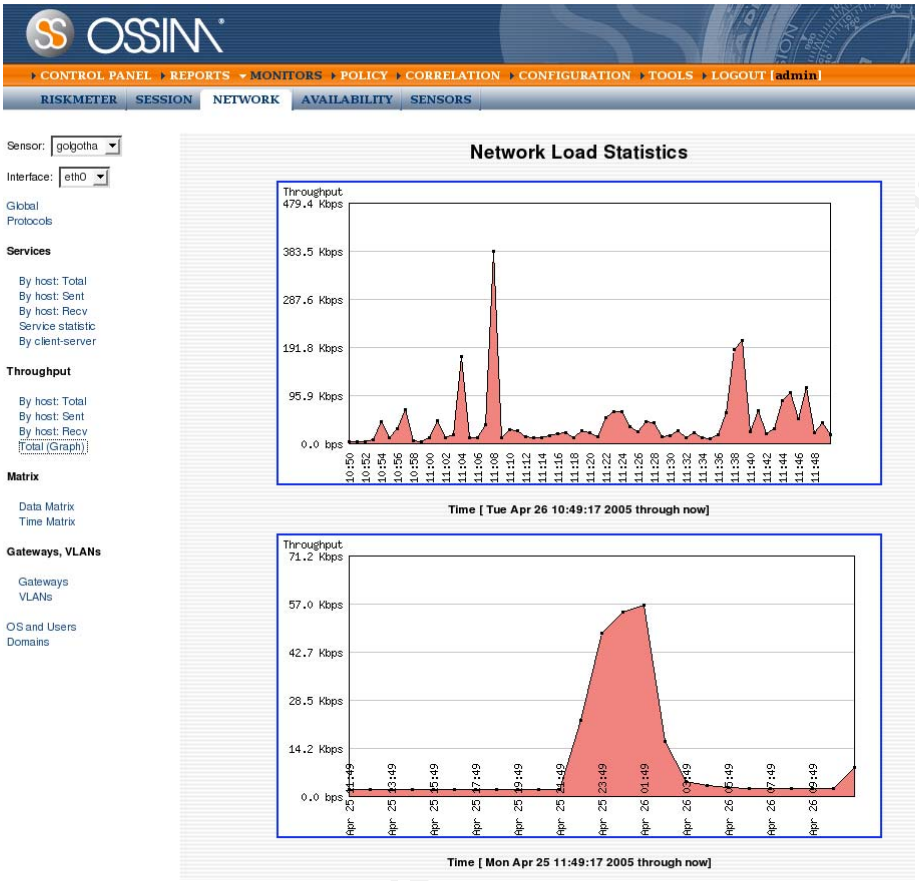
Network Traffic [TCP/IP]: Local Hosts - Data Sent+Received

Hosts: [All] [**Local Only**] [Remote Only]

Data: [**All**] [Sent Only] [Received Only]

Host	Domain	Data		FTP	HTTP	DNS	Telnet	NBios-IP	Mail	DHCP-BOOTP	SNMP	NNTP	NFS/AFS	X11	SSH	Gnutella	Kazaa	WinMX
192.168.2.175		155.1 MB	33.4 %	8.1 KB	11.8 MB	129.6 MB	0	276	75.8 KB	0	74	74	148	222	11.7 MB	74	0	0
golgotha	Local	154.7 MB	33.3 %	53.9 KB	111.6 MB	19.1 MB	0	0	0	0	0	0	15.3 MB	0	5.0 MB	0	0	0
[redacted]	Local	134.8 MB	29.0 %	0	7.6 KB	133.4 MB	0	988.9 KB	0	298.6 KB	0	0	0	0	0	0	0	0
[redacted]	Local	15.1 MB	3.2 %	0	0	14.8 MB	0	252.7 KB	0	0	0	0	0	0	0	0	0	0
wayreth	Local	4.3 MB	0.9 %	0	2.2 MB	0	0	0	0	0	0	0	0	0	787.0 KB	0	0	0
192.168.2.119		241.1 KB	0.1 %	0	0	0	0	241.1 KB	0	0	0	0	0	0	0	0	0	0
192.168.2.33		236.6 KB	0.0 %	0	356	0	0	233.5 KB	0	0	0	0	0	0	0	0	0	0
192.168.2.200		235.9 KB	0.0 %	0	444	141	0	235.1 KB	0	0	0	0	0	0	148	0	0	0
harpo	Local	230.7 KB	0.0 %	0	0	0	0	230.7 KB	0	0	0	0	0	0	0	0	0	0
memnon [NetBIOS]		68.6 KB	0.0 %	0	0	0	0	8.4 KB	0	0	0	0	0	0	0	0	0	0
midkemia	Local	17.8 KB	0.0 %	0	60	0	0	17.8 KB	0	0	0	0	0	0	0	0	0	0
d5qdl01j.[redacted]	Local	13.8 KB	0.0 %	0	0	0	0	12.4 KB	0	1.3 KB	0	0	0	0	0	0	0	0
carmen	Local	4.5 KB	0.0 %	0	0	0	0	4.5 KB	0	0	0	0	0	0	0	0	0	0
portatil-oscar	Local	3.6 KB	0.0 %	0	0	0	0	3.6 KB	0	0	0	0	0	0	0	0	0	0
port-icabrera	Local	1.7 KB	0.0 %	0	0	0	0	1.7 KB	0	0	0	0	0	0	0	0	0	0
srtapepis [NetBIOS]		1.2 KB	0.0 %	0	0	0	0	1.2 KB	0	0	0	0	0	0	0	0	0	0
jj [NetBIOS]		854	0.0 %	0	0	0	0	854	0	0	0	0	0	0	0	0	0	0
192.168.2.203		480	0.0 %	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
cosa2	Local	335	0.0 %	0	0	0	0	335	0	0	0	0	0	0	0	0	0	0
atrullas.[redacted]	Local	335	0.0 %	0	0	0	0	335	0	0	0	0	0	0	0	0	0	0
192.168.2.98		118	0.0 %	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The following diagram indicates load at particular times of the day on a network. These diagrams can indicate network abuse, misconfigurations, worms, or even normal use. These data points can be trended over time and compared to previous days/months for patterns.



Proprietary Detection/Prevention Tools:

There are many schools of thought when it comes to deciding between a capable open source tool and a proprietary tool. Regardless of your stance there are many proven tools that offer both commercial support and support from the open source community. Below we will highlight a few of these.

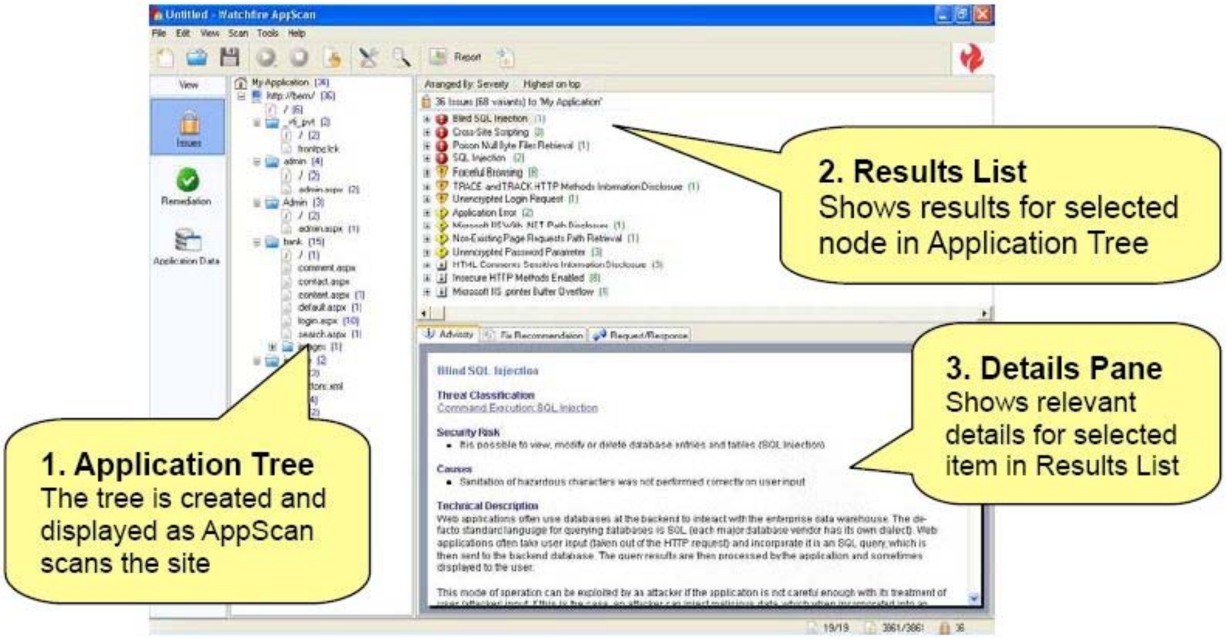
Watchfire AppScan

Whilst the majority of application scanning tools offer the ability to scan for coding vulnerabilities after an application has been built, Watchfire’s AppScan distinguishes itself in that it can be used throughout the software development lifecycle. It is not limited to use after an application has been completely built. The ability to leverage AppScan during the development lifecycle enables security conscious developers to fix security issues earlier in development thus saving considerable postfix dollars.

Some of the more common checks that AppScan checks for include cross site scripting, privilege escalation, session state monitoring, HTTP response splitting, parameter tampering, hidden field manipulation, backdoors/debug options, and buffer overflows. What differentiates AppScan from many of the other proprietary and open source tools available is the wide variety of useful reports that are generated. Within the AppScan tool remediation reports, executive dashboards, and compliance reporting are all offered. These features are increasingly valuable for larger more risk averse organizations which are subject to compliance, privacy, and regulatory standards.

AppScan is provided as a Windows based GUI which has been intuitively laid out. A logical directory structure is laid out from left to right. Issues, Remediation, and Application data adorn the left side and are featured main categories. From left to right the user may drill down successively

Figure 1: AppScan



Arranged by: Severity Highest on top

36 Issues (68 variants) to 'My Application'

- Blind SQL Injection (1)
- Cross-Site Scripting (3)
- Poison Null Byte Files Retrieval (1)
- SQL Injection (2)
- Forceful Browsing (8)
- TRACE and TRACK HTTP Methods Information Disclosure (1)
- Unencrypted Login Request (1)
- Application Error (2)
- Microsoft IIS With .NET Path Disclosure (1)
- Non-Existing Page Requests Path Retrieval (1)
- Unencrypted Password Parameter (3)
- HTML Comments Sensitive Information Disclosure (3)
- Insecure HTTP Methods Enabled (8)
- Microsoft IIS .printer Buffer Overflow (1)

Blind SQL Injection

Threat Classification
Command Execution: SQL Injection

Security Risk
- It is possible to view, modify or delete database entries and tables (SQL Injection)

Causes
- Sanitation of hazardous characters was not performed correctly on user input

Technical Description
Web applications often use databases at the backend to interact with the enterprise data warehouse. The de-facto standard language for querying databases is SQL (each major database vendor has its own dialect). Web applications often take user input (taken out of the HTTP request) and incorporate it in an SQL query, which is then sent to the backend database. The query results are then processed by the application and sometimes displayed to the user.

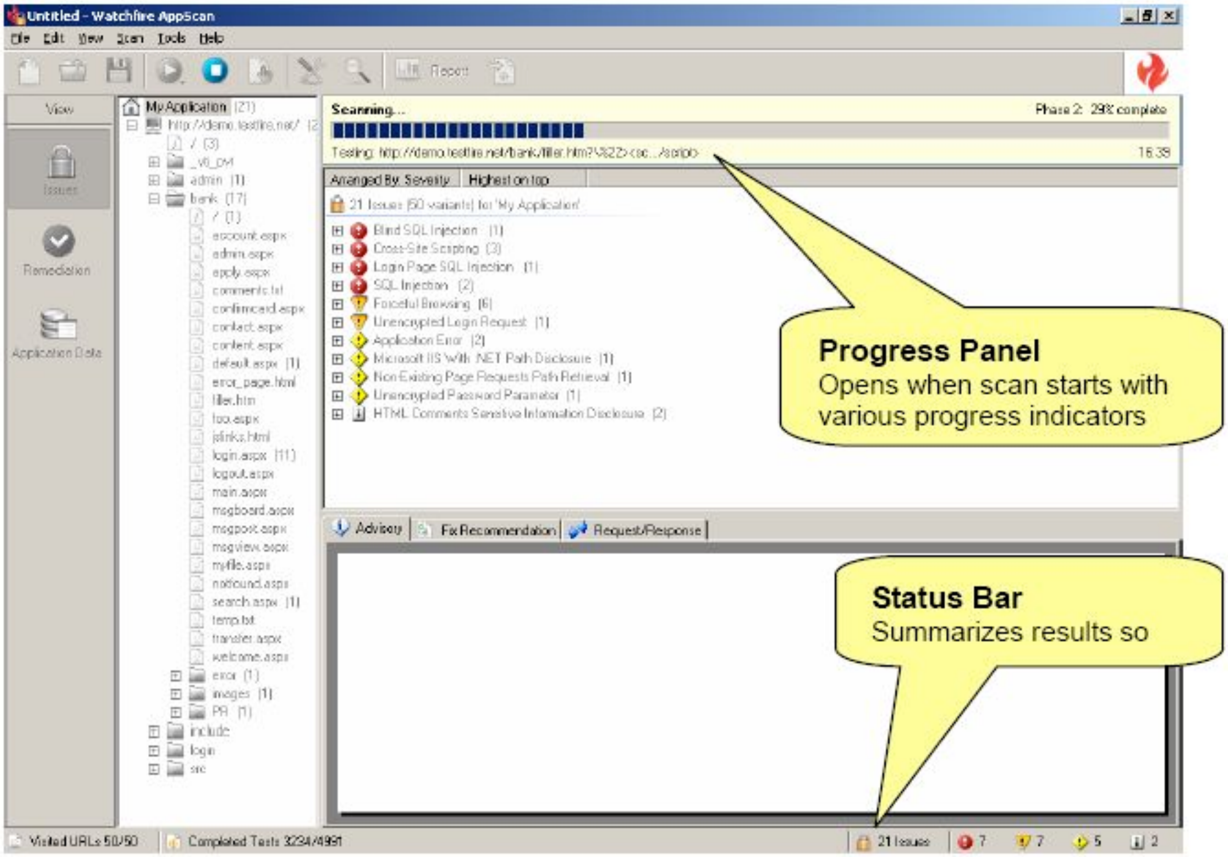
This mode of operation can be exploited by an attacker if the application is not careful enough with its treatment of

1. Application Tree
The tree is created and displayed as AppScan scans the site

2. Results List
Shows results for selected node in Application Tree

3. Details Pane
Shows relevant details for selected item in Results List

Figure 2: AppScan



```
My Application (21)
⊟ http://demo.testfire.net/ (2
    / (3)
  ⊞ _vti_cnf
  ⊞ admin (1)
  ⊟ bank (17)
      / (1)
      account.aspx
      admin.aspx
      apply.aspx
      comments.txt
      confirmcard.aspx
      contact.aspx
      content.aspx
      default.aspx (1)
      error_page.html
      filler.htm
      foo.aspx
      jsfinks.html
      login.aspx (11)
      logout.aspx
      main.aspx
      msgboard.aspx
      msgpost.aspx
      msgview.aspx
      myfile.aspx
      notfound.aspx
      search.aspx (1)
      temp.txt
      transfer.aspx
      welcome.aspx
    ⊞ error (1)
    ⊞ images (1)
    ⊞ PR (1)
  ⊞ include
  ⊞ login
  ⊞ src
```

```
Scanning...                                                        Phase 2: 29% complete
Testing: http://demo.testfire.net/bank/filler.htm?%22><sc.../script>             16:39
```

```
Arranged By: Severity | Highest on top
21 Issues (50 variants) for 'My Application'
⊞ Blind SQL Injection (1)
⊞ Cross-Site Scripting (3)
⊞ Login Page SQL Injection (1)
⊞ SQL Injection (2)
⊞ Forceful Browsing (6)
⊞ Unencrypted Login Request (1)
⊞ Application Error (2)
⊞ Microsoft IIS With .NET Path Disclosure (1)
⊞ Non Existing Page Requests Path Retrieval (1)
⊞ Unencrypted Password Parameter (1)
⊞ HTML Comments Sensitive Information Disclosure (2)
```

```
Visited URLs 50/50 | Completed Tests 3234/4991 | 21 Issues | 7 | 7 | 5 | 2
```

Progress Panel
Opens when scan starts with various progress indicators

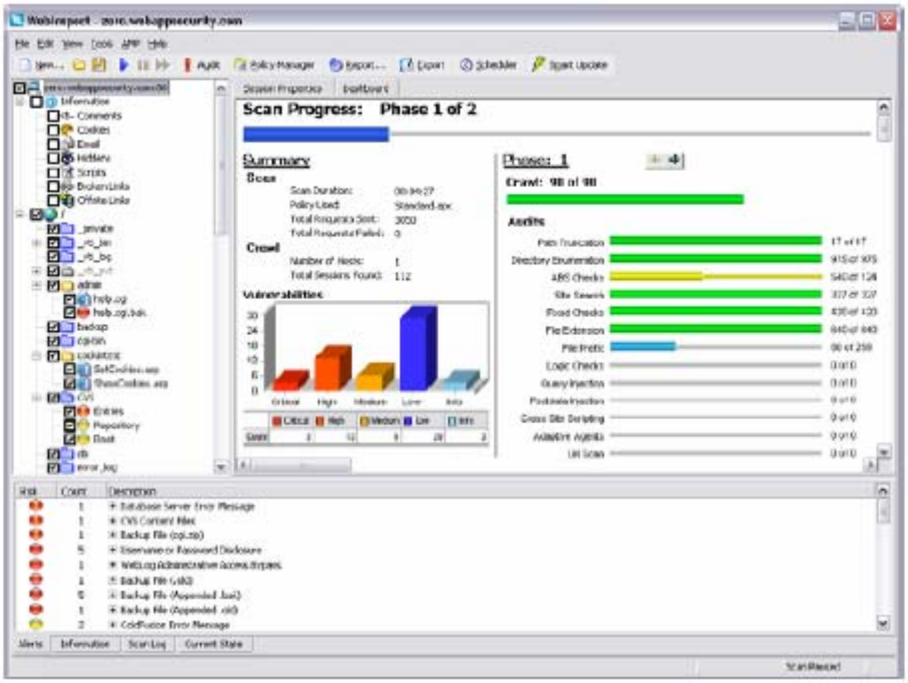
Status Bar
Summarizes results so

WebInspect

WebInspect is another extremely popular web application scanner amongst many security professionals. WebInspect application security assessment tools allow for the identification of known and unknown vulnerabilities, WebInspect can also check to ensure that a web application are secured properly, attempt common web attacks such as parameter substitution/injection, cross site scripting, directory traversal attacks, and much more.

WebInspect features a similar user interface as the one featured in AppScan. However, it does offer the ability to interact with the reporting tools in real time as scans are being performed. Interactive reports are also available which allow for drilling down into particular areas of concern.

Figure 3: WebInspect



Popular and useful pre-bundled tools are favorites with WebInspect. Amongst the offered tools are:

- A Cookie Cruncher which allows a security practitioner to assess the strength of the cookies used in browsing sessions.
- An SPI Proxy allows the security assessor to view each and every browser request and server response.
- WebBrute allows for brute forcing login forms.
- WebDiscovery allows for scanning of web services and correlates to them to specific ports for later investigation.

Both of these tools have been extremely popular amongst both security consulting firms and software development houses which provide web presence software. As for which one to use, both companies offer a free trial offer.

Open Source Prevention Tools:

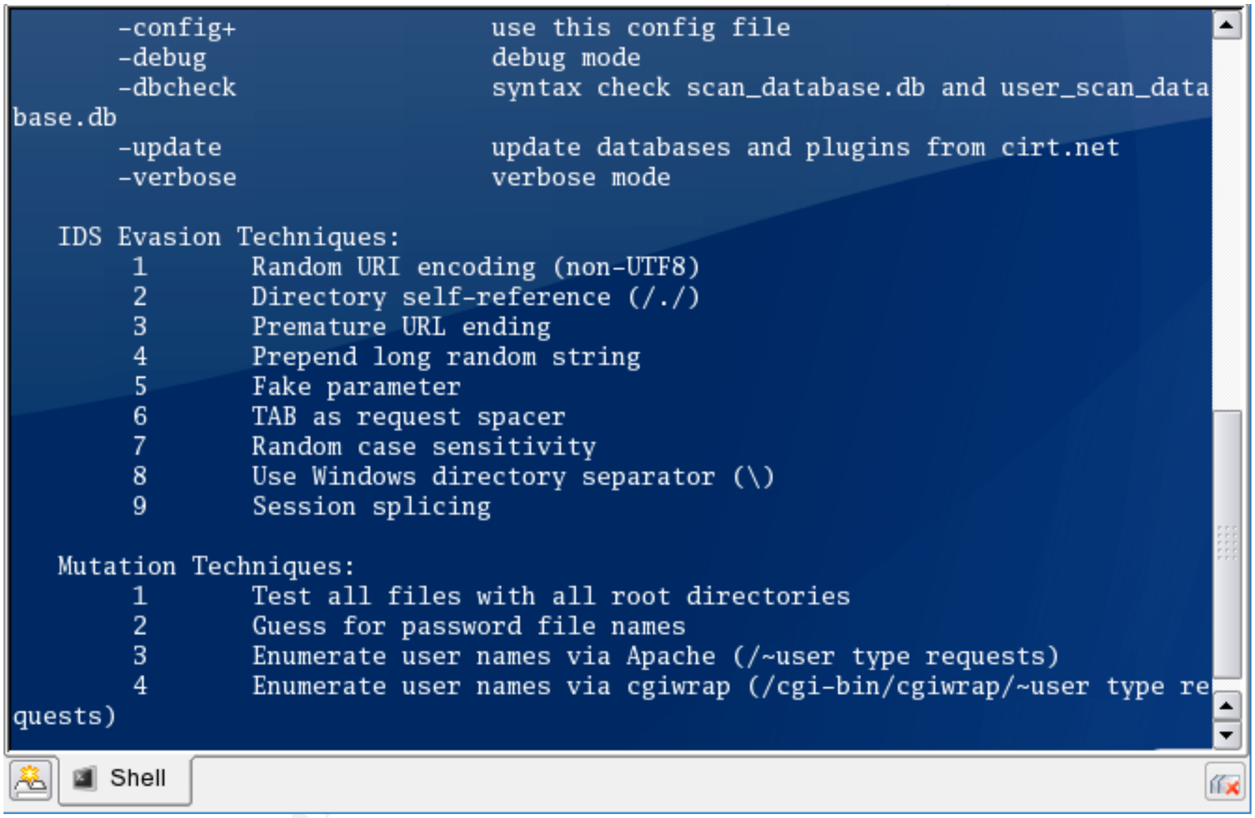
Nikto

Nikto is arguably the most popular web server scanner available in the open source community. Comprehensively scanning for over 3250 known dangerous files on as many as 600 servers Nikto has its name firmly imbedded amongst many seasoned security professionals as one of the premier web vulnerability scanners.

Nikto is a command line drive tool which features a wide variety of options including anti IDS or evasion scanning tactics. These are helpful when testing IDS deployments for attack signatures and IDS responses.

As with the proprietary tools mentioned earlier Nikto can be used as a preventative tool which can tip off developers of insecure configurations before critical web services are published to the World Wide Web or even to extranet partners. Nikto signature updates are similar to those features in many popular anti virus tools; however, they are updated by the open source community.

Figure 4: Nikto



```
       -config+                  use this config file
       -debug                    debug mode
       -dbcheck                  syntax check scan_database.db and user_scan_data
base.db
       -update                   update databases and plugins from cirt.net
       -verbose                  verbose mode

   IDS Evasion Techniques:
        1       Random URI encoding (non-UTF8)
        2       Directory self-reference (/./)
        3       Premature URL ending
        4       Prepend long random string
        5       Fake parameter
        6       TAB as request spacer
        7       Random case sensitivity
        8       Use Windows directory separator (\)
        9       Session splicing

   Mutation Techniques:
        1       Test all files with all root directories
        2       Guess for password file names
        3       Enumerate user names via Apache (/~user type requests)
        4       Enumerate user names via cgiwrap (/cgi-bin/cgiwrap/~user type re
quests)
```

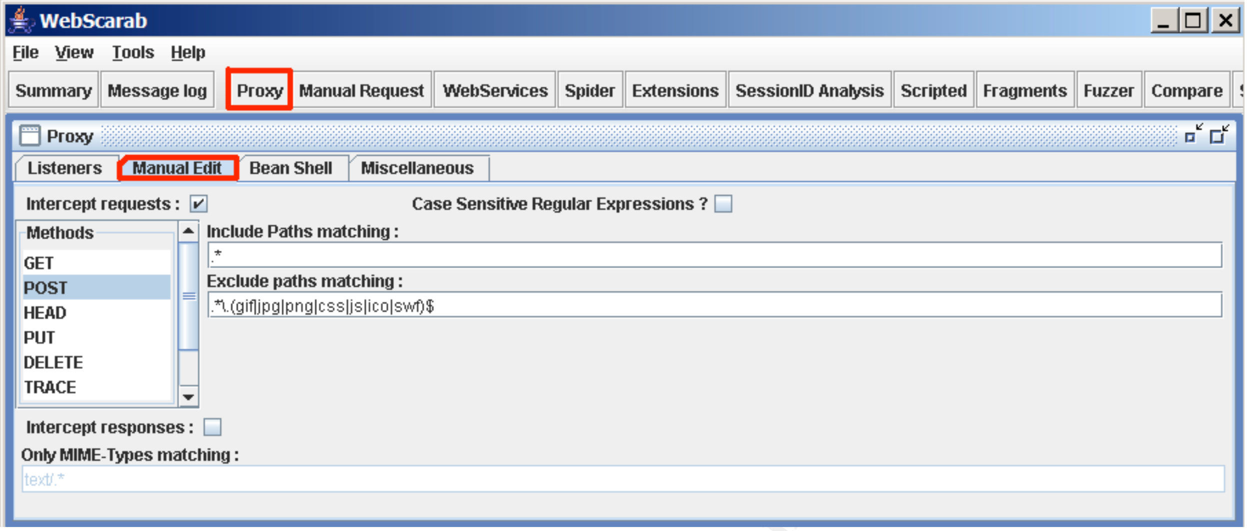
WebScarab

WebScarab is a java application which has been published by the OWASP project, an open source led community which we will speak in depth about later. The WebScarab framework is often used to dissect applications which communicate over HTTP or HTTPS. Typical uses include locating, identifying, and altering posts and requests made by a web browser before they are sent off to a web server. These options allow a user to slow down the communication between a web client and server so that requests and responses can be analyzed properly. Oftentimes during this analysis the improper handling of credentials, logins, cookies, etc are discovered thus allowing a user to subvert the security controls inherent within the web applications being analyzed. WebScarab is most commonly used by application developers who wish to more completely secure the web applications they are serving.

WebScarab features are plentiful, however the most commonly used include:

- **Proxy:** Able to observe both encrypted (HTTPS) and unencrypted traffic (HTTP). Encrypted traffic is viewed by establishing a second SSL tunnel between the proxy and the application web server.
- **SessionID Analysis:** a variety of cookies can be captured and analyzed to visually determine the randomness and unpredictability which should be inherent in all cookie generation.
- **Manual Request:** Editing and replay of previous requests to web applications can be viewed and manipulated.
- **Reveal Hidden Fields:** WebScarab allows a user to view hidden fields within the web presentation fields for manipulation, deletion, or reconnaissance.
- **Spider:** Allows for analysis on the web application which may reveal additional links that are not known initially.
- **Beanshell:** This is a feature which allows for the scripting of arbitrary java code on requests and or responses received by the web client or server.

Figure 5: WebScarab: Editing of web post



Intercept requests : ☑

Case Sensitive Regular Expressions ? □

Methods
GET
POST
HEAD
PUT
DELETE
TRACE

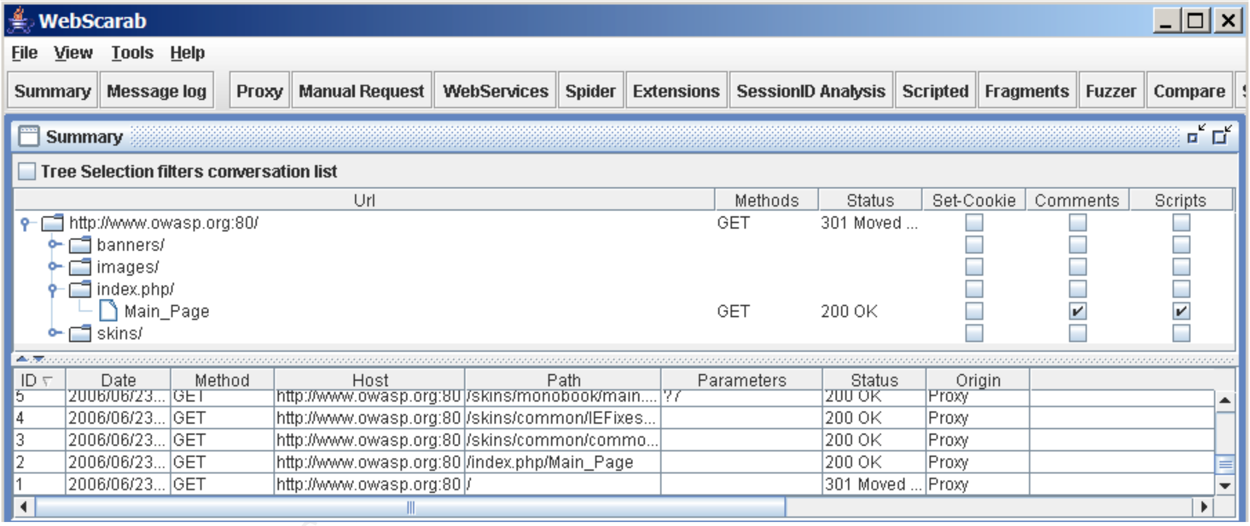
Include Paths matching : `.*`

Exclude paths matching : `.*\.(gif|jpg|png|css|js|ico|swf)$`

Intercept responses : □

Only MIME-Types matching : `text/.*`

Figure 6: WebScarab: Directory structure view of webpage



□ Tree Selection filters conversation list

Url	Methods	Status	Set-Cookie	Comments	Scripts
http://www.owasp.org:80/	GET	301 Moved ...	□	□	□
banners/			□	□	□
images/			□	□	□
index.php/			□	□	□
Main_Page	GET	200 OK	□	☑	☑
skins/			□	□	□

ID	Date	Method	Host	Path	Parameters	Status	Origin
5	2006/06/23...	GET	http://www.owasp.org:80	/skins/monobook/main....	?7	200 OK	Proxy
4	2006/06/23...	GET	http://www.owasp.org:80	/skins/common/IEFixes...		200 OK	Proxy
3	2006/06/23...	GET	http://www.owasp.org:80	/skins/common/commo...		200 OK	Proxy
2	2006/06/23...	GET	http://www.owasp.org:80	/index.php/Main_Page		200 OK	Proxy
1	2006/06/23...	GET	http://www.owasp.org:80	/		301 Moved ...	Proxy

Secure Coding

Awareness of the importance of web application security has jumped significantly in just a few years… 93 percent felt that secure application development was more of a priority now than three years ago(Symantec, 2006).

"Security training is at the heart of writing good code", writes John Heimann of Oracle(Heimann, 2006). For organizations that make available internet connected systems to the public for use, security training is a must have and is often overlooked in many developers backgrounds. "It's an unfortunate fact that most developers are not required to learn secure coding practices in school", noted Heimann. Too often within academia and within the corporate world the focus in educating developers is on creating efficient bug free code. Security checks are optional at best are rarely considered.

References to secure coding methodologies and practices can be found here:

http://www.microsoft.com/uk/msdn/security/developer_security.mspx

As an example, most programmers would consider the implementation of a specific check on the length of web form input to be unnecessary and wasteful of valuable CPU cycle times. Unfortunately for every programmer out there that is unaware of secure coding practices there are hackers out there that are not only aware but capable of searching for and taking advantage of these loopholes left behind by developers.

OWASP

The Open Web Application Security Project commonly referred to as OWASP is an open source project dedicated to the discovery and fighting of insecure software(Wikipedia).

The open source OWASP community is made up of corporations, educational entities, and developers and security practitioners located on all seven continents. They are best known for their creation of articles, methodologies, documentation, and tools which are all freely available for use the development and testing of secure application code(OWASP, 2007).

Some of the notable tools that OWASP has produced include Web Goat, a web application penetration testing environment, WebScarab, mentioned earlier, and the OWASP.NET tools set. Most famously though OWASP is responsible for producing the OWASP Top Ten, which is an application security awareness document. In the next section we will highlight the Top Ten, and the vulnerabilities most often found within web applications.

OWASP – Top Ten

The OWASP Top Ten is often considered the defacto standard when looking at many audit controls and security firms responsible for assessing code.

Specifically PCI or the payment card industry security standard, a standard enforced jointly between American Express, Visa, MasterCard, and JCB requires organizations which store or process credit card data to implement secure coding procedures within their SDLC.

This requirement ensures that all web applications developed in house are securely coded and reviewed for the following vulnerabilities(PCI and Data Security Compliance, 2007).

Top Ten Highlights

Before getting into a few specific types of attack we will dive into some of the more prominent top ten vulnerabilities from a higher level. It is important to understand the breadth of attacks that are covered by the top ten to truly understand its value amongst the development community.

Unvalidated Input:

Unvalidated Input is defined as any information received via web requests which are not validated before being used by a web application.

Commonly attackers target these vulnerabilities to attack backend components through a web application. Usual targets include the web application layer itself, the underlying OS that supports the web application and backend databases which often contain sensitive customer information, credit cards, social security numbers, and intellectual property.

Examples of unvalidated input include cross site scripting attacks, buffer overflows, and injection flaws.

Buffer overflows are also common areas of concern when writing secure applications. Exploiting buffer overflows simply requires an attacker to identify fields in a vulnerable application which do not perform bounds checking. Writing outside the bounds of a memory block can corrupt data, crash the program, and even crash the entire operating system subjecting an organization to unnecessary downtime.

Broken Access Control:

Another commonly overlooked vulnerability when creating web applications are broken access controls. Access is granted to specific authorized users of web applications to perform a specific function based upon their need and general role that they serve. Broken access control vulnerabilities provide a means for malicious users to escalate privileges beyond their intended permissions. This flaw may allow for the unauthorized viewing of other users accounts, the viewing of sensitive configuration files, or the escalation to that of a root user.

Broken Authentication and Session Management

In the realm of online banking the sanctity of the user authorization/authentication component is paramount to the both the customers and the banks well being and reputation. Without a tested and proven safe authentication model customers are unlikely to use online banking, something that saves banks significant amounts of money annually (Lee, 2001).

Authentication and session management relate to the direct processing of user, application, or system authentication and the subsequent need to manage these active sessions. Without reliable authentication, trust is lost and accountability escapes the process.

Authentication methods with web applications nearly always involve a username and password. Because of this it is vitally important to ensure that these credentials are not easily accessible. Features such as password resets and forgotten usernames should all be protected and require authentication via a password hint, mothers' middle name, or some other authentication method.

Commonly overlooked areas of weakness within web applications include:

1. Password Strength – ensure that PAM modules or like enforcement is used within the application. Dictionary word attacks are common amongst web applications, and unless complexity is enforced authentication mechanisms are
2. Password Use – Too many failed authentication requests in a given period of time should lock or freeze an account for a given period of time.

3. Password Change Controls – When changing a password users should be required to provide both the old and new password before being able to change a password. Use of email as a confirmation of a password change is another enforcement mechanism which will often provide an alert to an unsuspecting user of someone tampering with their login information.
4. Browser caching – neither Authentication nor session data should be presented in retrieval requests from web servers. In addition cache options within a web application may allow a browser to store cookies. This can be dangerous for computers which are accessible from internet cafes, libraries, and other publicly accessible terminals. Caching of cookies could allow another unauthorized user to purport another user through cookie/session id manipulation.

Injection Flaws

Injection Flaws are used to subvert the required authentication and authorization process to an Operating System, application, or other service. Malicious users can take advantage of weaknesses in applications which fail to filter client based commands and make system calls, use external applications such as netcat, ssh, and even run entire scripts. These weaknesses can be present anytime an interpreter is used within a web application; as such code must be reviewed before pushing out to production systems.

SQL injection flaws are some of the most popular attack vectors for hackers as they can often reveal enormous amounts of sensitive data. Customer information, credit card numbers, employee information, social security numbers, and other sensitive information.

SQL Injection Attack Example

As previously mentioned SQL injection attacks can occur when user input on the client side is not filtered properly. Escape characters are maliciously used to break away from the currently command and initiate a new listening state on the underlying web application. Once the web application is in a new listening state a new oftentimes unauthorized SQL query against a back end database can be performed. Table views, dropping tables, and adding usernames are all very popular methods of gaining unauthorized access to both applications and underling stored data.

The following example illustrates this:

An attacker who wishes to retrieve the usernames of all active users could do so with an application that uses an SQL backend and one that does not filter SQL requests as the application or middleware layer. For this particular example a user is wishing to learn about the usernames authorized to use an application.

Within the domain mybank.com the following can be used to retrieve usernames provided the table names that are passed are accurate. Many times it is easy to guess the table names that are used to identify parameters such as username, DOB, SOCSEC, password, etc.

Example:

User visits: http://www.mybank.com

When attempting to retrieve the usernames of fellow bankers http://www.mybank.com/login/login.asp?loginid=bobsmith or d=d

The generated SQL statement is because of this passed argument is:

SELECT loginid, FirstName FROM User where loginid = bobsmith or d=d

This condition will always be true because d always equals d. Thus all loginid and FirstName pairs will be returned (Imperva, 2007).

Another clever way to manipulate SQL injection attacks is to DROP tables which can cause irreparable harm to an underlying database and supported application.

http://www.mybank.com/login/login.asp?loginid=bobsmith;DROP TABLE USERS

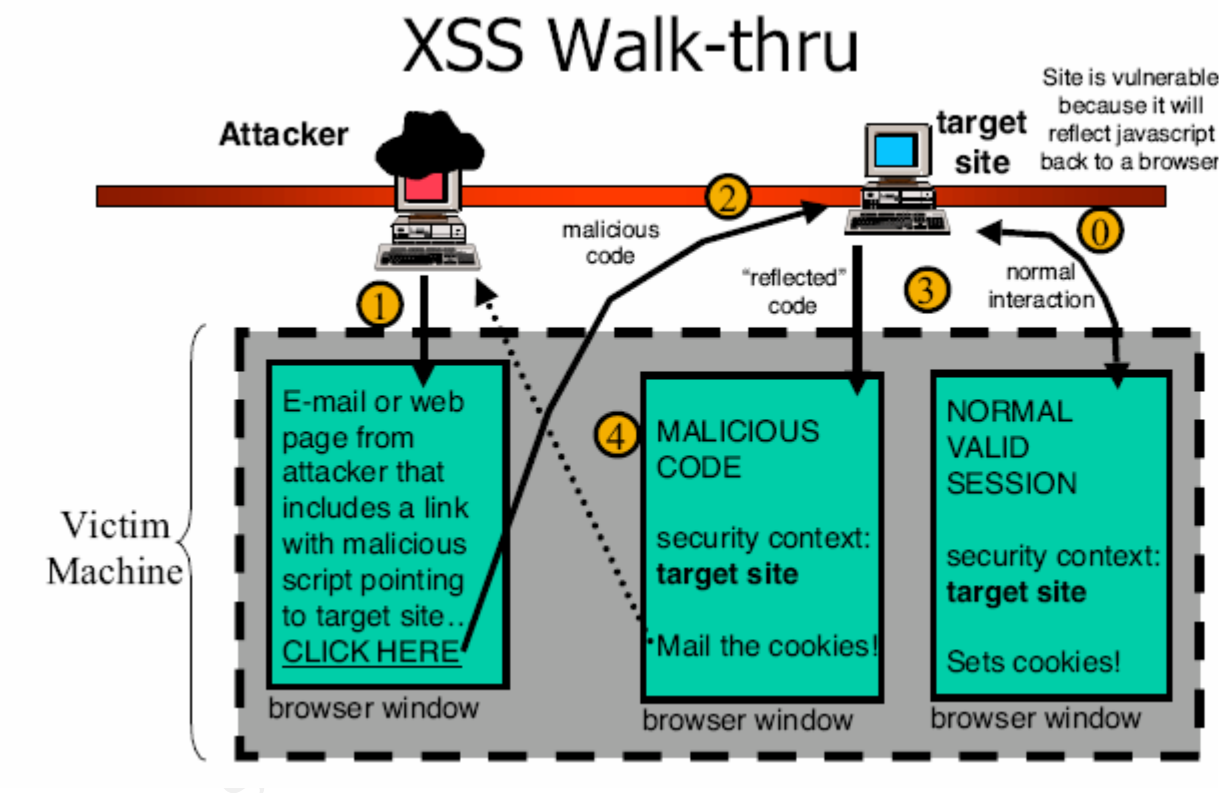
This statement will delete the USERS table provided it exists. Deleting this USERS table would likely render the application useless as users would not be able to login to the underlying application(SQLCourse.com, 2006).

SQL injection attacks are only limited to the creativity and resourcefulness of the attacker. Usernames could be added along with passwords, billing amounts could be changed, interest paid, balances changed are all possible with weak filtering at the application and database layer.

Unvalidated Input Example: XSS

Cross site scripting is an attack targeted towards the hosting web application, underlying OS, and often backend database. An attacker will often attack web applications that do not filter scripts from form fields submitted to web applications. For example, attackers are often able to insert code which gets executed by the user's browser. This code will attempt to steal browser cookies that might include banking session data, passwords, or the like. Session cookies are then used by the attacker to emulate a legitimate user session to a banking site, email account, or the like.

The diagram below illustrates this relationship between attacker, web application, and website using a XSS attack.


XSS Walk-thru
Attacker
target site
Site is vulnerable because it will reflect javascript back to a browser
0
1
2
3
4
malicious code
"reflected" code
normal interaction
Victim Machine
E-mail or web page from attacker that includes a link with malicious script pointing to target site.. CLICK HERE
browser window
MALICIOUS CODE
security context: target site
Mail the cookies!
browser window
NORMAL VALID SESSION
security context: target site
Sets cookies!
browser window

(Skoudis, 2005)

Getting into the technical jargon behind some of the advanced cross site scripting attacks is beyond the scope of this paper.

A fantastic series of examples of cross site scripting can be found here: http://www.devshed.com/c/a/Security/A-Quick-Look-at-Cross-Site-Scripting/1/

Summary

Many of these attacks can be prevented and or detected before they do irreparable harm. Proper defenses require a defense in depth approach. Only a few of the possible approaches have been discussed in this paper, and likely in time these strategies will require tweaking and improvement as attack vectors will invariably evolve over time.

In summary a combination of application of both technologies and user awareness are the only effective ways of truly defending against web attacks. Technologies such as application layer firewalls, reverse proxies, Intrusion Detection and Prevention systems coupled with a solid security training program for application developers will yield significant security enhancements. Additionally the use of code review tools and scanners will provide proactive resources that both developers and in house security professionals can leverage in discovering application layer weaknesses.

For many businesses which conduct business online, their reputation is at stake. One breach can oftentimes lead to irreparable brand damage. And putting a price on the amount of damage done is oftentimes extremely difficult, though losses to public companies can be in excess of billions when stock valuations are considered.

http://esj.com/Case_Study/article.aspx?EditorialsID=2249

REFERENCES

Desmond, Paul (2004, May 17). All-out blitz against Web app Attacks

Retrieved December 30, 2006, from networkworld.com

Web site: (1)

http://www.networkworld.com/techinsider/2004/0517techinsidermain.html

Heimann, John (2006, May 23). The Importance of Security Training.

Retrieved October 31, 2006, from CIO Update website:

http://www.cioupdate.com/article.php/3608391

Imperva, (2007, February 1) SQL Injection.

Retrieved February 10, 2007 from: http://www.imperva.com/application_defense_center/glossary/sql_injection.html

Internet World Stats. (2007, January 11). Internet World Stats*, Usage and Population Statistics*.

Retrieved February 01, 2006, from http://www.internetworldstats.com/stats2.htm

Krebs, Brian (2006, September 28). ID Thieves Turn Sights on Smaller E-Businesss. Retrieved December 30, 2006, from WashingtonPost.com

Web site: (1)

http://www.washingtonpost.com/wp-dyn/content/article/2006/09/28/AR2006092800333.html

Gartner (2006, May 12). Magic Quadrant for Security Information and Event Management.

Retrieved February 28, 2007, from Novell.com

Web site: (1)

http://www.novell.com/products/sentinel/gartner.pdf

Gartner (2005, May 2). Improve IT Security with Vulnerability Management

Retrieved February 27, 2007, from Gartner.com

Web site: http://www.gartner.com/DisplayDocument?doc_cd=127481

Halloway, Jason (No Date Provided). Risk and Security Rewards.

Retrieved February 27, 2007, from CSOONLINE.COM

Web site: (1)

http://www.csoonline.com/caveat/012907.html

Lee,Mie-Yun (2001, July 1). Money in the eBank – training customers to trust online banks

Retrieved February 10,2007, from http://findarticles.com/p/articles/mi_m0DTI/is_7_29/ai_79826907

OWASP, (2007, February 10) Welcome to OWASP.

Retrieved February 10, 2007 from:

http://www.owasp.org/index.php/Main_Page

OWASP. (2006, December 21). In *Wikipedia, The Free Encyclopedia*.

Retrieved February 01, 2007, from

http://en.wikipedia.org/wiki/OWASP

PCI and Data Security Compliance. (2007, January 19). Wordpress website

Retrieved February 15, 2006, from http://datasecurity.wordpress.com/2007/02/05/owasp-top-10-for-2007/

Skoudis, Ed (2005). SANS TRAINING TRACK 4.

Retrieved October 31, 2006, from SANS Institute training documents.
Not distributed publicly.

SQLCourse.com. (2006, May 12). SQLCourse.com website

Retrieved October 31, 2006, from

http://sqlcourse.com/drop.html

Symantec News Release. (2006, September 19). Symantec.com website

Retrieved October 31, 2006, from
http://www.symantec.com/about/news/release/article.jsp?prid=20060919_01

Last Updated: June 28th, 2009

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS WhatWorks Summit in Forensics and Incident Response	**Washington, DC**	**Jul 06, 2009 - Jul 14, 2009**	**Live Event**
SANS Singapore 2009	**Singapore, Singapore**	**Jul 06, 2009 - Jul 11, 2009**	**Live Event**
SANS Rocky Mountain 2009	**Denver, CO**	**Jul 07, 2009 - Jul 13, 2009**	**Live Event**
SANS SOS London 2009	**London, United Kingdom**	**Jul 13, 2009 - Jul 18, 2009**	**Live Event**
SANS Future Visions 2009 Tokyo	**Tokyo, Japan**	**Jul 15, 2009 - Jul 17, 2009**	**Live Event**
AUD439: Auditing OCTAVE Debut	**Washington, DC**	**Jul 20, 2009 - Jul 21, 2009**	**Live Event**
SANS IMPACT 2009	**Kuala Lumpur, Malaysia**	**Jul 27, 2009 - Aug 01, 2009**	**Live Event**
SANS SEC563: Mobile Device Forensics Debut	**Baltimore, MD**	**Jul 27, 2009 - Jul 31, 2009**	**Live Event**
SANS Boston 2009	**Boston, MA**	**Aug 02, 2009 - Aug 09, 2009**	**Live Event**
SANS Atlanta 2009	**Atlanta, GA**	**Aug 17, 2009 - Aug 28, 2009**	**Live Event**
SANS WhatWorks in Virtualization and Cloud Computing Security Summit 2009	**Washington, DC**	**Aug 17, 2009 - Aug 21, 2009**	**Live Event**
SANS Virginia Beach 2009	**Virginia Beach, VA**	**Aug 28, 2009 - Sep 04, 2009**	**Live Event**
SANS SCDP SEC556: Comprehensive Packet Analysis - Sept. 2009	**Ottawa, ON**	**Sep 09, 2009 - Sep 10, 2009**	**Live Event**
SANS Critical Infrastructure Protection at Oceania CACS2009	**Canberra, Australia**	**Sep 10, 2009 - Sep 11, 2009**	**Live Event**
SANS Network Security 2009	**San Diego, CA**	**Sep 14, 2009 - Sep 22, 2009**	**Live Event**
SANS SCDP Cutting Edge Hacking Techniques - June 2009	**Ottawa, ON**	**Sep 15, 2009 - Sep 15, 2009**	**Live Event**
SANS Canberra 2009	**OnlineAustralia**	**Jun 29, 2009 - Jul 04, 2009**	**Live Event**
SANS OnDemand	**Books & MP3s Only**	**Anytime**	**Self Paced**