Global Information Assurance Certification Paper

A comparative study of attacks against Corporate IIS and Apache Web Servers

GIAC (GCFW) Gold Certification

Author: Craig S Wright, craig.wright@information-defense.com
Advisor: Antonios Atlasis

Accepted: 20 July 2011

Abstract

It has been suggested that an attacker[1] will specifically target the Windows operating system. This research has shown that rather than this being the case an attacker will in fact not target Microsoft Windows, but rather seeks to avoid attacking Linux. This study has shown significant support for the assertion that an attacker shies away from Linux and not that they aim to attack windows. In particular, this study was designed to collect data on attacks against corporate Web servers. Unlike many of the other proceeding studies using honeypot systems, this experiment was designed to collect information on attacks against "*secured*" corporate systems.

[1] The Term "Attacker" has been used in this paper to refer to the commonly used designations of "hacker" or "cracker" and covers anyone attacking the computer host.

1 Introduction

It has been suggested that Microsoft Server Software is more likely to be attacked than Linux (Broersma, 2005) due to perceived insecurities within these systems. Previous research has focused on investigating the trends[2] against the underlying operating system as a whole (Honeynet Project & Research Alliance, 2005b, 2005a). The purpose of this research was to investigate a single factor, namely, the Web server software as a vector for attack.

This project was not designed to test the relative strengths or security levels of either Operating System, but rather to determine the relative attractiveness of each of these systems (the system being the combination of the web server and the underlying O/S) to an attacker.

In this experiment, the systems were configured to appear as a financial services organisations client website. There were two systems, one running on Apache and the other on IIS. All other services have been firewalled and only access to the web server using HTTP (TCP port 80) and HTTPS (TCP port 443 over SSL) was allowed. The actual pages and data presented by the web servers were identical but for the server headers. Unlike previous research, the focus of the experiment was to record the volume of attacks from individual sources. By this process, we were able to answer the following questions:

1. Which web server software product (IIS or Apache) will have the most "vulnerability scans"[3]?
2. Which product will an attacker spend the most time with in an attempt to "break" or "crack" the software security settings?

[2] The "*Know your enemy – Trend analysis*" from "The HoneyNet Project" has spawned several side projects and more in-depth honeynetworks designed to analyze and record attacks (Honeynet 2004a, 2004b).

[3] A Scan is defined as a single attempt to gain information on the system from a single host for the purpose of this experiment. This includes fingerprinting the application as can be done using either a vulnerability scanner or another tool that analyses the service version.

3. What is the subsequent effect of hiding the host headers[4] on the servers?

2 Methodology used in the study

The research was based on a controlled trial with naturally randomized subjects. The honeypot design without notification allowed for the randomised discovery of the systems used in the tests. This naturally excluded targeted attacks from this study as the systems:

1. Only simulated real banking sites.
2. Had not been available for a sufficient amount of time to be replicated in search engines.

A *robots.txt* file excluding all search engines and the Internet Archive was implemented to act as if the system was not allowing caching. This was added to stop Google scans and similar intelligence gathering methods as these could bias the experiment. This was designed in part to cover the fact that the real system was not available before the start of the experiment.

By restricting access to the servers through a firewall to only the Web service on TCP ports 80 and 443 it was possible to demonstrate system attractiveness on a single defined service. The results of this experiment support the research efforts of the HoneyNet Project (HoneyNet Project & Research, 2005), Symantec (Symantec, 2004), and the Internet Storm Centre (The SANS Institute, 2005). In correlation to the prior research on this topic it was initially confirmed that a greater number of attacks were made against the Windows server.

Rather than focus on the survivability of a host, this experiment has been designed to determine the attractiveness of the host to an attacker. Unlike many of the experiments on this topic, which have preceded this one, the experiment has been designed to test the effect of obscuring the servers by hiding the host headers and information thus available to an attacker.

[4] This is the HTTP Server host header. Both IIS and Apache allow an administrator to change or "hide" this system field.

The interesting effect shown in this study was not that IIS on Windows was more attractive than Apache on Linux, but rather that Linux is less attractive to attackers. The reasons for this have not been determined conclusively, but it would seem that LAMP5 based systems where less attractive than IIS, MSSQL and .Net based implementations. For the purpose of this paper we have not tested the effects of Apache on Windows which would make an interesting follow-up test.

The amount of time and effort an attacker spends on a particular system cannot be used to determine the difficulty in attacking that system. Attractiveness is not the same as survivability and a follow-up test was conducted using a perceived vulnerability in both Windows and LAMP.

The follow-up test involved configuring the system to appear as if it had vulnerability as defined by a Nessus scan. A collection of IIS 7 and Apache vulnerabilities were simulated. These vulnerabilities were selected randomly from the CVE list and were simulated using HPING (see appendix 5) and Snort (see the Snort Manual[6]) to modify the packets returned by both servers such that they mirrored the responses of a vulnerable system. The vulnerable versions of the software were not used as it would not be possible to control for individual responses (most vulnerable software versions can be attacked in a number of ways). This was used to differentiate attacks from Worms and Users. The traffic was filtered using SNORT to both record the data as well as to determine and monitor attacks.

SNORT allows for the creation of automated packet responses. Using the React function defined in section 2.11.4 of the SNORT Manual, a honey pot style web page can be fashioned for just the selected attack making the attacker believe they have discovered a vulnerable system.

For instance, in the alert rule for cve.2009-4444:

---

5 Linux, Apache, MySQL, PHP

6 This is covered in Section 2.11.4 (React) of the SNORT manual (from http://www.snort.org/assets/166/snort_manual.pdf). In place of a default 403 forbidden page, a "DEENIABLE_REACT" can be configured in the rules using a simulated vulnerable page. So when an alert is generated for an attack, it can be used to create a special web page as a honeypot.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"ET
WEB_SERVER Possible Microsoft Internet Information Services (IIS) .asp
Filename Extension Parsing File Upload Security Bypass Attempt (asp)";
flow:established,to_server; content:".asp|3B 2E|"; fast_pattern:only;
nocase; http_uri; classtype:web-application-attack; sid:2010592;
rev:8;)
```

Will be altered to add a "react" statement as follows:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"Customed
message associated with the expected response...)";
flow:established,to_server; content:".asp|3B 2E|"; fast_pattern:only;
nocase; http_uri; classtype:web-application-attack; sid:2010592; rev:9;
react: block;)
```

The block statement stops the page hitting the original target and send it to either a single default page or other alternate pages if these are configured.

To enable this, SNORT needs to be built with the following build option (also, the `--flexresp compile option is required`):

```
./configure --enable-react / -DENABLE_REACT
```

Then, the page can be configured to read from a file.

```
config react: <block.html>
```

or in our case to read from a customised file which is configured to return the same format as the attacker would obtain if attacking the site successfully:

```
config react: <file_loaded.html>
```

When configuring this option, it is critical that the file must contain the entire response, including any HTTP headers. Images and other information used in building the fake page can be located on an alternative server and the "block" page should appear as the error being modelled[7].

Note that the file must contain the entire response, including any HTTP headers. But, this response is not stringently limited to HTTP. It is also possible to craft a binary payload of arbitrary content if this is the expected response from the attack.

[7] A separate paper on the creation of these pages, Inline redirection and the use of HPING3 to replay packets and to create a SNORT Honeypot is being developed and written by the author of this paper and should be ready soon after this paper is loaded and published. This paper will also detail the use of IPTables and Snort Snarf to redirect packets on the fly.

Further, a "%s" is used in the rules to load the SNORT "msg" extension to the react rule and this can be used to load customised responses based on details about the selected attack, data called in the attack and more. Here, the "react" statement would be entered as:

```
react: block, msg;)
```

Here the defaul message in snort also need to be changed from:

```
<default_msg> ::= \
        "You are attempting to access a forbidden site.<br />" \
        "Consult your system administrator for details.";
```

To something more appropriate to the response that the attack would be expected to elicit.

2.1 Description of experimental study

Subjects (i.e. the attackers) discovered the systems based on their own activities. As there is no way to attract Subjects to the systems, it was expected that a random sample of the population of "hackers" on the Internet would find and explore the system at any given time. No details of the source of the attacks were analysed. These details (including attack sources, IP addresses and browser header information) have been recorded for later analysis.

The analysis took the nature of the probes into account as well as the relative amount of time spent on each system.

3 Experimental procedure

The data collected from this experiment is based on two "*honeynets*" deployed to appear as the primary and Disaster Recovery (DR) site for a fictitious financial organisation. Each HoneyNet consisted of two servers configured to run VMWare. Each of the VMWare servers was configured to run a Web server using bridged network access.

The first virtual server was configured using Red Hat Enterprise Linux 6.0 and Apache version 2.2. The second virtual server consisted of a Windows 2008 server

system running IIS 7.0. Each of the pages was configured to appear as a client portal in our fictitious financial services organisation (Figure 1).

Figure 1 - The "Staff Credit Union" banking login page

This organisation was created through an amalgamation of existing Credit Union and Banking sites and was designed to look and feel as "real" as possible. The domain was not formally registered, but a DNS zone was created on the local server. This zone and domain were never published nor were they advertised.

To simulate a financial services organisation, these systems were installed behind a firewall, which only allowed access to TCP port 80 and a simulated access to TCP port 443 when authenticated. The HoneyNet was linked to "*real*"[8] servers from a fictitious organisation. Both Systems were configured to require authenticated access before allowing any access to a backend application.

[8] These systems will be running the software being tested though they will have no real function.

Using the Snort IDS[9] software, the number of attacks and thus the effort expended by an attacker on each server was measured and recorded. The open source IDS product, SNORT[10], was used to collect the data. SNORT was installed on the underlying operating system that hosted the virtual machines as well as running the IPTables firewalls. A separate monitoring server was run on a Spanned switch port. The underlying system was Redhat Enterprise Linux 6.0. The Redhat system was configured with an IP address that could not be routed to the Internet and which was monitored for access (no access or compromise was noted in this system). In this manner, it was not possible to detect the IDS deployment from the web server. All systems where patched fully before being deployed (See Appendix 2 for statistics on the patching of deployed systems).

There were two phases to the first stage of the experiment with the first phase involving leaving the web host headers unaltered. The second phase involved hiding the web host headers. By this, the server was stopped from advertising its software version. In the first phase, the systems responded as Apache version 2.2 and IIS 7.0. In the second phase, both systems where configured to respond with "Secure Web Server version 9.3".

No other changes were made and a determined attacker using P0F or NMap could have differentiated the IP packet information to determine a difference between a Linux or Windows system (as a guess).

The results were collected daily for the period of the test[11]. Informational data was excluded from the results. All attacks detected by SNORT were collected together and no effort has been made to correlate the levels of attack against each server. The Monitoring server was used to collect all network traffic to and from the servers. The switch was spanned on the Monitoring Server port and TcpDump was set to capture all traffic. This was set as both a backup for the Snort systems as well as means of collecting any attacks that had been made against systems that could have bypassed Snort.

[9] IDS – Intrusion Detection System
[10] For details on Snort and to be able to download and register see Http://www.snort.org
[11] The test was conducted from February 2010 to December 2010.

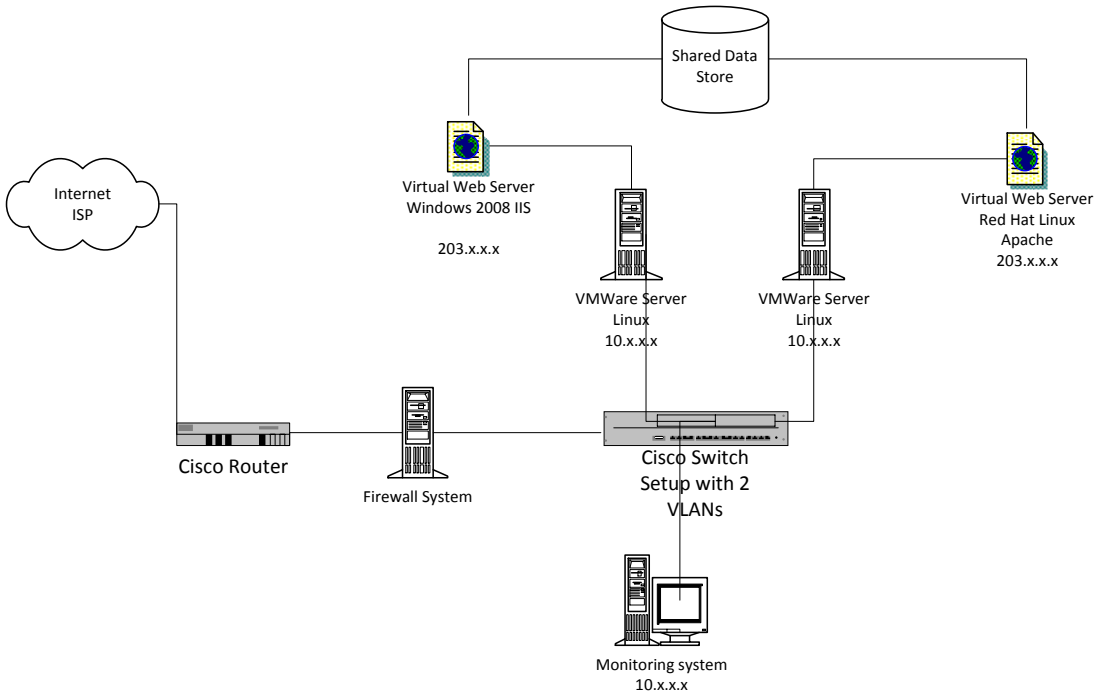


Figure 2 - Network Diagram

Future research and analysis to correlate the levels of attacks and statistically analyse the intensity of attacks against each server as a function of time is planned.

3.1 Steps to physically control variation

In order to minimise variation, the HoneyNet Servers were configured as follows;

- Both systems were installed on matching hardware and domain addresses,
- Both systems were booted from a Live DVD based created using RHEL in the manner of a Knoppix distribution.
- Both systems resided on the same switched network and be active at the same times.
- The IDS system was not be available or visible to the external network.
- Results were randomized as the systems were not "*advertised*" and it is expected that they were by general network scans and probes.

- The IP[12] addresses of the systems were sequentially allocated such that a probe could detect both at the same time.

3.2 Steps to statistically control variation

When either system was attacked by a DoS or DDoS[13] attack, both systems were made unavailable using IPTables. Snort was configured to trip a default block rule that took effect on both systems for any IP address that was noted as attempting a DoS or DDoS by Snort. This was done as the test was not designed to determine DoS/DDoS situations and it was decided that it was better to not continue to record data on one system while the other is not being tested.

The Honeypots were deployed using the methodology detailed in the paper by Greg M. Bednarski and Jake Branson (2004) titled, "*Information Warfare: Understanding Network Threats through Honeypot Deployment*".

4 Results and Discussion

We subsequently analysed the data based on the average number of individual source hosts and attacking each system and the number of individual attacks registered per host. If an attacker had been attacking from multiple systems (such as from the use of a Botnet), this was not determined and each attacking host was included individually.

The test was stopped following the collection of data for phase 1 of the test and the systems were no longer routed for a period of six weeks before a new IP address in the same C-Class range was allocated and the system was redeployed using a simulated and randomly selected vulnerability from the list of vulnerabilities. Vulnerabilities were selected randomly from the CVE list (CVE[14]). The selected vulnerabilities used are included in Appendix 1.

[12] Internet Protocol
[13] Denial of Services (DDOS is a Distributed Denial of Services) attack.
[14] The CVE list is available online at http://cve.mitre.org/data/downloads/allcves.html

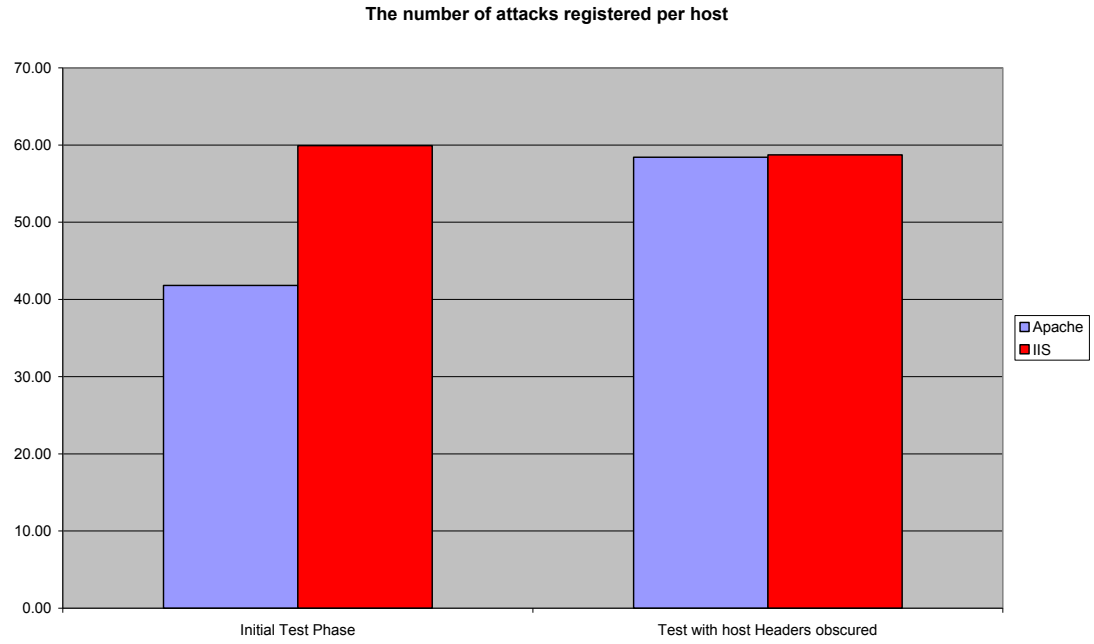


Figure 3 Phase 1 results of attacks against IIS and Apache (attacks / day)[15]

The systems were left running with the traffic being recorded to collect data for several weeks with the Web servers host headers remaining visible (Phase 1). Next, we reconfigured the servers to each display an alternate host header; "*Secure Web Server version 2.3*" (Phase 2). This was designed to obscure the system details from a potential attacker.[16]

IIS		Apache	
Medium	**High**	**Medium**	**High**
1090.1	2097.0	1242.5	1272.9

Table 1 Mean time spent attacking with Vulnerabilities (seconds)

The results of these tests have been displayed in figure 3. From the results we can say that there are no statistical differences between attacks against the hosts when the host headers are obscured. In Table 1, the mean amount of time an attacker spent in attacking the web service is displayed for CVE vulnerabilities. The high and medium

[15] See appendix 3 for the definitions of Attacks used in this document.

[16] Definitions used for terms such as an Attack within this document are included in Appendix 3 below.

level ratings are determined using the CVSS[17] rating (http://www.first.org/cvss/).

In phase two, the average time per attacking IP address was recorded for both the simulated medium and high level attacks. Again we see a clear distinction between IIS and Apache with attackers spending more time on IIS high level attacks (2097.0 seconds) than Apache high level attacks (1272.9 seconds). Phases are separated depending on if the server headers have been changed or not[18]

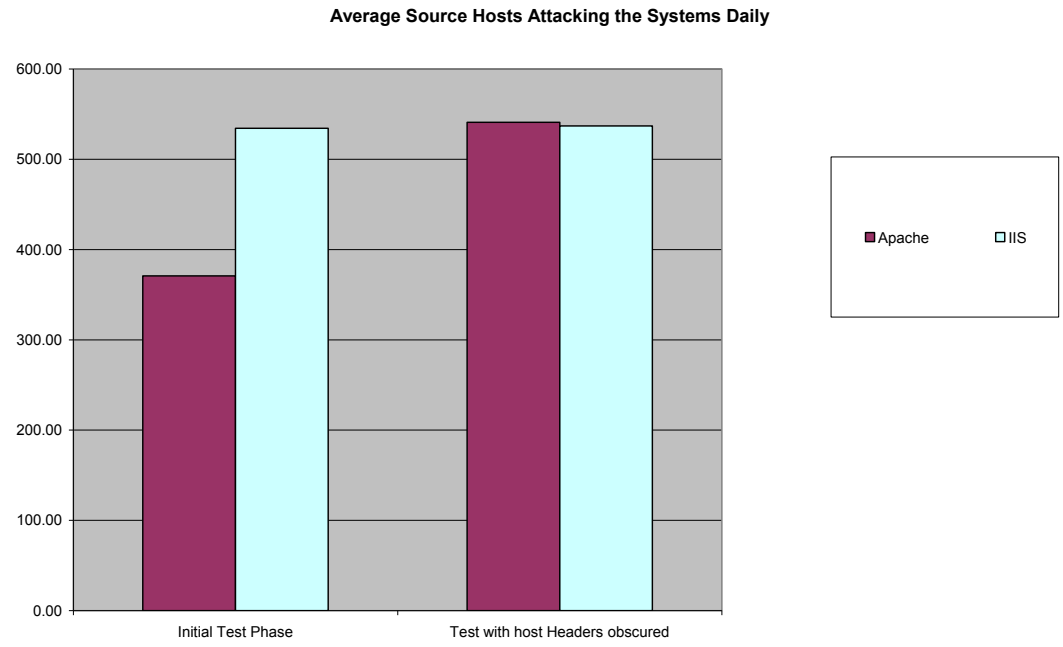


Figure 4 Phase 1 results of attacks against IIS and Apache

From these results we can see that an attacker will extend significant effort against an IIS based system with a perceived high level vulnerability.

It would also be possible to posit an alternative explanation that the mean time does not necessarily shows a preference, but also "*difficulty*" on exploiting the target. No evidence has been found that attacking IIS is significantly more difficult than attacking

[17] Further information on this rating system is available online. For a detailed description of this rating system http://scap.nist.gov/events/2010/itsac/presentations/day1/SCAP_101-CVE_and_CVSS.pdf. These documents are not directly referenced in this paper, but serve as a valuable introduction to this rating system.

[18] The two phases of data collection are separated depending on if the server headers have been changed or not. Phase 1 is set using the default host headers for the web server, Phase 2 involved changing to the alternate host header; "Secure Web Server version 2.3".

Apache (or for that matter that Linux is less difficult than Windows to attack) and it is more likely that the difficulty of an attack is related to the individual vulnerability and not the underlying system.

In addition, the economics of an attack would likely lead to seeking the easier target if this was the case. In such an event, it would be logical to posit that given a choice of a Linux or Windows based system, if Windows was "*more difficult*" to attack, a clear preference for Linux would be expressed with a lower time being spent on Windows systems expressing the same data.

4.1 Apache is less attractive to attackers

The results of the experiment clearly demonstrate a similarity in the results obtained when the server type either is unknown or is determined to be a Microsoft Windows system. However, there was a markedly lower intensity and volume of attacks against the Apache Web server when its host headers were displayed.

Mean Daily Results	**Apache**		**IIS**	
	Number of attacks detected per host	**Number of Source Hosts Detected Per Day**	**Number of attacks detected per host**	**Number of Source Hosts Detected Per Day**
Initial Test Phase	41.82	370.71	59.93	534.36
Test with host Headers obscured	58.43	540.86	59.54	536.93

Table 2 Mean attacks by day

In order to determine whether the Microsoft Windows IIS Web server or the Apache Linux Web server would attract a larger number of scans or attacks than its counterpart, a two-sample t-test[19] was performed on the "*Number of Source Hosts Detected per Day*" (**Figure**). When choosing a null hypothesis (H_o) that there is no difference between the Apache or IIS Web server, it was found that the results of the initial phase of the experiment were significantly different ($t = 29.59$, $df = 54$, $p <$

[19] See appendix 4 for a detailed description of the statistical tests used in this analysis.

0.1507) at the alpha = 20 level[20]. Therefore we rejected the Null and accepted the alternative hypothesis H_A, that there was indeed a difference between the servers.

From the results there is evidence to support the hypothesis that the Apache Web server on Linux is less likely to be attacked then IIS Web server on Microsoft Windows 2008.

4.1.1 Attacks against Linux with Apache are less intense

An ANOVA[21] analysis of the results of the subsequent tests test demonstrate a significant difference ($F=5.4402$; $df = 3$, $p < 0.0019$) in the intensity of the attacks. It can clearly be seen in **Figure** that an attacker stops an attack with less effort when it has been determined that they are attacking Apache on Linux.

A comparison of the number of attacks detected per host for both the obscured IIS server, and either server with the host headers altered demonstrated no significant difference ($F=0.0007$; $df = 2$, $p=0.9993$).

Again we reject the null hypothesis (H_o) that there is no difference between the server groups tested. Means Comparisons for all pairs using Tukey-Kramer HSD at alpha = 5 show that Apache Web servers are less likely to be attacked.

4.1.2 Attackers treat unknown web servers as IIS

ANOVA Analysis of Attacks by source hosts when the header was not Apache on Linux[22][23] demonstrated no significant difference ($F=0.0007$; $df = 2$; $p=0.9993$).

ANOVA again supports the assertion that there is no significant variation ($F=0.0344$; $p=0.8538$, RSquare = 0.000859) when we compare the results of the phase 1 tests against IIS to the phase 2 tests with the host headers obscured[24].

Conversely an analysis by ANOVA of the phase 1 tests against Apache to the

[20] A higher level of alpha was chosen for the initial test as a lower volume of data had been collected at this point.
[21] A number of statistical processes (including ANOVA) are detailed in Appendix 4.
[22] See Appendix 4.
[23] The results were either an unknown server (based on the hidden host header) or the IIS web service.
[24] The two phases of data collection are separated depending on if the server headers have been changed or not. Phase 1 is set using the default host headers for the web server, Phase 2 involved changing to the alternate host header; "Secure Web Server version 2.3".

phase 2 tests with the host headers obscured significantly (F=5.7659; p=0.0211, df =3) supports our claim that attackers are less likely to attack Apache on Linux.

Further, when these results are coupled with the initial analysis (F=14.4513; p=0.0004, df = 3) of attacks against Apache vs. IIS from above it is easy to see that there is support for the assertion that an attacker does not care what the server is as long as it is not Linux.[25]

These results would suggest that the threat against Internet deployed hosts is moving from automated scanning tools to more human intensive processes. By specifically avoiding the Apache Linux system (when not obscured), there is evidence to support the contention that attackers are manually targeting systems and actively stopping attacks they deem to be "*too difficult*".

5 Limitations in this study

No effort was made to analyse and the levels of attacks against any server. It may be that more high-level attacks are made against a Linux server for example; this assertion has not been tested. In this study, all levels of attack were treated equally whether they were designated as a low, medium or high-level attack.

As noted above, it would also be possible to posit an alternative explanation that the mean time does not necessarily shows a preference, but also "*difficulty*" on exploiting the target. It would be necessary to create a separate experiment in order to collect evidence as to whether attacking IIS is significantly more difficult than attacking Apache (or for that matter that Linux is less difficult than Windows to attack) and it is more likely that the difficulty of an attack is related to the individual vulnerability and not the underlying system.

6 Conclusions and Future Work

Some potential areas of further research have emerged from this study. It is clear

[25] H_0, There are no differences between the number of attacks against a server type;
H_A, There is a difference between at least one of the tests. Tests conducted at the alpha = 5 level.

that an attacker will avoid Linux servers that are not obscured, though this study can provide no reasons for this behaviour. A further and interesting test would be to use an Apache installed on a Windows server. That could be used to show if attackers avoid Apache or Linux.

It is suggested that researchers consider this study and its conclusions as an initial exploration into the methodology of an attacker. Research into the motivations driving this behaviour in an attacker needs to be determined. Further research is essential in order to develop appropriate strategies and measures to secure systems sufficiently. It is essential to understand the psychology of the attacker if effective controls are to be developed and deployed.

A study where the host headers on a Microsoft Windows IIS host are altered to simulate Apache on Linux could determine some further important results.

This study has shown that attackers are not so much attracted to Windows, but rather shy away from Linux based systems.

One potential reason for this could be the increased W32 market penetration. Another possible reason could stem from a perceived greater level of security with Linux hosts. The results of this study do not demonstrate that either Linux or Microsoft Windows is more secure. However, the results do support the claim that attackers believe that Linux is more difficult to attack, as it is more secure. As the average attacker was willing to spend a greater amount of effort in an attempt to compromise a high vulnerability (Table 1) IIS Windows system when defined by the amount of time spent attacking the system, we can see that IIS has become a more attractive target for attackers than LAMP based systems running the same data.

Further research is needed on this topic to determine "*WHY*" Linux is less attractive than Windows to attackers. In addition, experiments into the effects of using other systems (such as the MAC OS) could be further explored.

Author Name, email@address

7 Appendices

7.1 A list of the used vulnerabilities used

The following appendix contains a list of vulnerabilities used in the experiment.

7.1.1 Apache Vulnerabilities

CVE-2010-4172 Multiple cross-site scripting (XSS) vulnerabilities in the Manager application in Apache Tomcat 6.0.12 through 6.0.29 and 7.0.0 through 7.0.4 allow remote attackers to inject arbitrary web script or HTML via the (1) orderBy or (2) sort parameter to sessionsList.jsp, or unspecified input to (3) sessionDetail.jsp or (4) java/org/apache/catalina/manager/JspHelper.java, related to use of untrusted web applications.

CVE-2010-2068 mod_proxy_http.c in mod_proxy_http in the Apache HTTP Server 2.2.9 through 2.2.15, 2.3.4-alpha, and 2.3.5-alpha on Windows, NetWare, and OS/2, in certain configurations involving proxy worker pools, does not properly detect timeouts, which allows remote attackers to obtain a potentially sensitive response intended for a different client in opportunistic circumstances via a normal HTTP request.

CVE-2010-1587 The Jetty ResourceHandler in Apache ActiveMQ 5.x before 5.3.2 and 5.4.x before 5.4.0 allows remote attackers to read JSP source code via a // (slash slash) initial substring in a URI for (1) admin/index.jsp, (2) admin/queues.jsp, or (3) admin/topics.jsp.

CVE-2010-1452 The (1) mod_cache and (2) mod_dav modules in the Apache HTTP Server 2.2.x before 2.2.16 allow remote attackers to cause a denial of service (process crash) via a request that lacks a path.

CVE-2010-0009 Apache CouchDB 0.8.0 through 0.10.1 allows remote attackers to obtain sensitive information by measuring the completion time of operations that verify (1) hashes or (2) passwords.

CVE-2009-3250 The saveForwardAttachments procedure in the Compose Mail functionality in vtiger CRM 5.0.4 allows remote authenticated users to execute arbitrary code by composing an e-mail message with an attachment filename ending in

(1) .php in installations based on certain Apache HTTP Server configurations, (2) .php. on Windows, or (3) .php/ on Linux, and then making a direct request to a certain pathname under storage/.

CVE-2009-2412 Multiple integer overflows in the Apache Portable Runtime (APR) library and the Apache Portable Utility library (aka APR-util) 0.9.x and 1.3.x allow remote attackers to cause a denial of service (application crash) or possibly execute arbitrary code via vectors that trigger crafted calls to the (1) allocator_alloc or (2) apr_palloc function in memory/unix/apr_pools.c in APR; or crafted calls to the (3) apr_rmm_malloc, (4) apr_rmm_calloc, or (5) apr_rmm_realloc function in misc/apr_rmm.c in APR-util; leading to buffer overflows. NOTE: some of these details are obtained from third party information.

CVE-2009-1195 The Apache HTTP Server 2.2.11 and earlier 2.2 versions does not properly handle Options=IncludesNOEXEC in the AllowOverride directive, which allows local users to gain privileges by configuring (1) Options Includes, (2) Options +Includes, or (3) Options +IncludesNOEXEC in a .htaccess file, and then inserting an exec element in a .shtml file.

CVE-2008-7271 Multiple cross-site scripting (XSS) vulnerabilities in the Help Contents web application (aka the Help Server) in Eclipse IDE, possibly 3.3.2, allow remote attackers to inject arbitrary web script or HTML via (1) the searchWord parameter to help/advanced/searchView.jsp or (2) the workingSet parameter in an add action to help/advanced/workingSetManager.jsp, a different issue than CVE-2010-4647.

CVE-2008-0455 Cross-site scripting (XSS) vulnerability in the mod_negotiation module in the Apache HTTP Server 2.2.6 and earlier in the 2.2.x series, 2.0.61 and earlier in the 2.0.x series, and 1.3.39 and earlier in the 1.3.x series allows remote authenticated users to inject arbitrary web script or HTML by uploading a file with a name containing XSS sequences and a file extension, which leads to injection within a (1) "406 Not Acceptable" or (2) "300 Multiple Choices" HTTP response when the extension is omitted in a request for the file.

7.1.2 IIS Vulnerabilities

CVE-2010-3229 The Secure Channel (aka SChannel) security package in Microsoft Windows Vista SP1 and SP2, Windows Server 2008 Gold, SP2, and R2, and Windows 7, when IIS 7.x is used, does not properly process client certificates during SSL and TLS handshakes, which allows remote attackers to cause a denial of service (LSASS outage and reboot) via a crafted packet, aka "TLSv1 Denial of Service Vulnerability."

CVE-2010-1886 Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP1 and SP2, Windows Server 2008 SP2 and R2, and Windows 7 allow local users to gain privileges by leveraging access to a process with NetworkService credentials, as demonstrated by TAPI Server, SQL Server, and IIS processes, and related to the Windows Service Isolation feature. NOTE: the vendor states that privilege escalation from NetworkService to LocalSystem does not cross a "security boundary."

CVE-2010-0112 Multiple SQL injection vulnerabilities in the Administrative Interface in the IIS extension in Symantec IM Manager before 8.4.16 allow remote attackers to execute arbitrary SQL commands via (1) the rdReport parameter to rdpageimlogic.aspx, related to the sGetDefinition function in rdServer.dll, and SQL statements contained within a certain report file; (2) unspecified parameters in a DetailReportGroup (aka DetailReportGroup.lgx) action to rdpageimlogic.aspx; the (3) selclause, (4) whereTrendTimeClause, (5) TrendTypeForReport, (6) whereProtocolClause, or (7) groupClause parameter in a SummaryReportGroup (aka SummaryReportGroup.lgx) action to rdpageimlogic.aspx; the (8) loginTimeStamp, (9) dbo, (10) dateDiffParam, or (11) whereClause parameter in a LoggedInUsers (aka LoggedInUSers.lgx) action to (a) rdpageimlogic.aspx or (b) rdPage.aspx; the (12) selclause, (13) whereTrendTimeClause, (14) TrendTypeForReport, (15) whereProtocolClause, or (16) groupClause parameter to rdpageimlogic.aspx; (17) the groupList parameter to IMAdminReportTrendFormRun.asp; or (18) the email parameter to IMAdminScheduleReport.asp.

CVE-2006-5858 Adobe ColdFusion MX 7 through 7.0.2, and JRun 4, when run on Microsoft IIS, allows remote attackers to read arbitrary files, list directories, or

read source code via a double URL-encoded NULL byte in a ColdFusion filename, such as a CFM file.

CVE-2010-1256 Unspecified vulnerability in Microsoft IIS 6.0, 7.0, and 7.5, when Extended Protection for Authentication is enabled, allows remote authenticated users to execute arbitrary code via unknown vectors related to "token checking" that trigger memory corruption, aka "IIS Authentication Memory Corruption Vulnerability."

CVE-2010-0112 Multiple SQL injection vulnerabilities in the Administrative Interface in the IIS extension in Symantec IM Manager before 8.4.16 allow remote attackers to execute arbitrary SQL commands via (1) the rdReport parameter to rdpageimlogic.aspx, related to the sGetDefinition function in rdServer.dll, and SQL statements contained within a certain report file; (2) unspecified parameters in a DetailReportGroup (aka DetailReportGroup.lgx) action to rdpageimlogic.aspx; the (3) selclause, (4) whereTrendTimeClause, (5) TrendTypeForReport, (6) whereProtocolClause, or (7) groupClause parameter in a SummaryReportGroup (aka SummaryReportGroup.lgx) action to rdpageimlogic.aspx; the (8) loginTimeStamp, (9) dbo, (10) dateDiffParam, or (11) whereClause parameter in a LoggedInUsers (aka LoggedInUSers.lgx) action to (a) rdpageimlogic.aspx or (b) rdPage.aspx; the (12) selclause, (13) whereTrendTimeClause, (14) TrendTypeForReport, (15) whereProtocolClause, or (16) groupClause parameter to rdpageimlogic.aspx; (17) the groupList parameter to IMAdminReportTrendFormRun.asp; or (18) the email parameter to IMAdminScheduleReport.asp.

CVE-2009-4890 Multiple cross-site scripting (XSS) vulnerabilities in the login application in vBook 4.2.17 allow remote attackers to inject arbitrary web script or HTML via the (1) title and (2) message parameters.

CVE-2009-4445 Microsoft Internet Information Services (IIS), when used in conjunction with unspecified third-party upload applications, allows remote attackers to create empty files with arbitrary extensions via a filename containing an initial extension followed by a : (colon) and a safe extension, as demonstrated by an upload of a .asp:.jpg file that results in creation of an empty .asp file, related to support for the NTFS Alternate Data Streams (ADS) filename syntax. NOTE: it could be argued that this is a

vulnerability in the third-party product, not IIS, because the third-party product should be applying its extension restrictions to the portion of the filename before the colon.

CVE-2009-4444 Microsoft Internet Information Services (IIS) 5.x and 6.x uses only the portion of a filename before a ; (semicolon) character to determine the file extension, which allows remote attackers to bypass intended extension restrictions of third-party upload applications via a filename with a (1) .asp, (2) .cer, or (3) .asa first extension, followed by a semicolon and a safe extension, as demonstrated by the use of asp.dll to handle a .asp;.jpg file.

7.2 Patching Statistics in real systems.

The following table (Table 3) has been extracted from Wright and Tanveer (2011).

Table 3. Patching Analysis of Audited Systems.

	No. Analyzed	95% Confidence Interval of days between patching (Mean)	Average Policy Patch time (CI)	% Prior Reports noting patching
Windows Server	1571	41.1, 122.4 (86.2)	55.5, 87.9	98.4%
Windows Client	13,951	22.8, 69.3 (48.1)	29.6, 49.4	96.6%
Other Windows Applications	30,290	58.1, 181.8 (125.2)	68.1% NA	18.15%
Internet Facing Routers	515	58.2, 164.1 (114.2)	58.1% NA	8.7%
Internal Routers	1,323	129.3, 384.6 (267.8)	73.2 NA	3.99%
Internal Switches	452	139.9, 483.9 (341.2)	87.5 NA	1.2%
Firewalls	1,562	21.5, 65.7 (45.4)	24.5, 108.2	70.7%

Note that in a test of patching and security configurations of systems that native Windows applications and servers are patched far more than are the supporting devices or the applications running on the server.

7.3 Attack Level Definitions

These are the definitions associated with network and host based attacks.

7.3.1 Critical

Any systems compromise is a Critical attack. Critical events include:

- A system compromise is any attack that has gained unauthorised access (including altering of files on the respective system).
- Bypassing a firewall filter or other security controls (inc VLANS) when this is not permitted.
- Any DOS {Denial of Service} (including DDOS) attack that significantly impairs performance.
- Virus infections or Trojans that are not stopped and infect systems.

7.3.2 High Security Risk

Attacks with the potential to effect or compromise a system.

These are appropriate or targeted attacks. High level risks are those that concern relevant attacks against relevant systems and security controls. These are issues that need to be addressed as soon as possible to stop them becoming a critical issue. Any high level attack has the potential to become a critical event on a system if left unattended.

7.3.3 Medium Security Risk

Skilled scans or attacks with the potential to affect the system if security controls (including patching) were not in place. These are targeted but filtered attacks.

A medium level attack is defined as one that is targeted towards the systems in place but is not likely to succeed due to other factors that are in place. An example of this would be an attack against a patched web server. The attack may be listed as high if the system was unpatched, but is now unlikely to cause any noticeable effect.

7.3.4 Low Security Risk

A low level attack is an attack with little or no likelihood of compromising a

system. These are often general probes and tools often run by unsophisticated attackers.

An example of a low level attack would be an attacker running an IIS targeted attack tool against an Apache web server on Linux. The attack being directed towards a Microsoft Web server running IIS is not likely to cause any noticeable issues on a Linux based system with Apache. There are exceptions to this, for example, if that version of Apache was configured with FrontPage extensions, than this attack (if against Windows ASP extensions) could be relevant and may be thus classified as either High or Medium.

7.3.5 Suspicious Activity

Suspicious Activity covers all traffic and system behaviour that is not explainable or does not conform to any reasonable expectation of an attack and is not capable of causing damage to the system.

7.3.6 Modifiers

The following events are modifiers and may affect the level of an attack as reported.

High volume of attacks

If a high volume of a particular attack occurs, the severity level may be increased. An example of this is:

Attack	Low volume	High Volume
SCAN XMAS	Low Level Attack	Medium Level Attack
ICMP Source Quench	Low Level Attack	Medium Level Attack
WEB-MISC Attempt to execute cmd	Medium Level Attack	High Level Attack

In the examples above, the volume affects the level assigned to the attack as a large number of packets consumes bandwidth and may affect performance. In the Web example, a large volume of attacks from a single source may signify a new or unpatched vulnerability that the attacker is trying to exploit and thus needs to be investigated.

Skilled and/or unexpected attacks

"ICMP Source Quench" is generally considered a *Suspicious* packet and not an attack. If these packets have been forged or it is suspected that a "trusted" host has been compromised to send these, the attack may be rated as either *Low* or even *Medium*.

An example of this would be if "ICMP redirect host" packets where being received from the ISP upstream router.

7.3.7 Definition matrix

The following table is a guide for determining levels of risk associated with an attack.

	Critical	High	Medium	Low	Suspicious
Denial of Service Attack (DOS or DDOS)	Current and continuing loss of service	Possible loss of service if action is not taken	Service could be slightly effected if the attack was to ensue	No loss of service likely to occur	ICMP or large traffic amounts that are unlikely to affect service
Interactive System level compromise	Compromised systems or evidence of such an attempt				
Unauthorized file access/ modification	Compromised systems or evidence of such an attempt	Suspicion of or attempts to access to protected files			
Blocked attacks as noted on the Firewall	Packets that are bypassing the installed firewall policy	Evidence of packet shaping / detailed spoofing in order to bypass firewall rules	Packets targeted at a specific service that may be vulnerable from other sites	General scans	Misc dropped packets
Attacks as noted on the DMZ IDS hosts	System vulnerable to this attack	Targeted attacks on an open service (esp. if recently patched)	Detailed probes and continuing scans against specific services	General Scans	
Virus or Worm attacks	Systems infected	Evidence of a virus or worm passing the Anti-virus system	New virus or worm detected	Virus or worm blocked on external anti-virus server	

7.4 Statistical Tests

These are the statistical tests used in this analysis.

two-sample t-test[26]

The two-sample t-test is used in comparing two independent sample means (t-test) with Homogeneity of Variance

This test is used to compare two sample means. The independent variable is nominal level data and the dependent variable is interval/ratio level.

Analysis of Variance (Anova)

Analysis of variance is merely regression when the predictive variables are qualitative. Covariance analysis is regression with some qualitative predictive variables and some quantitative predictive variables (the latter are then called "covariates" or "covariables").

The easiest way to understand analysis of variance is as a generalization of Student's t-test: it informs the user if the mean of a quantitative variable is the same in several groups (Student's t-test is limited to the case of two groups) or, in other words, if a quantitative variables depends on a qualitative variable.

A more general way of understanding analysis of variance is as a test comparing two models: checking if a quantitative variable y depends on a qualitative variable x is equivalent to comparing the models y ~ x and y ~ 1 (if the two models are significantly different, the more complex one, y ~ x, brings more information, i.e., the quantitative variable y depends on the qualitative variable x, i.e., the mean of y is not the same in the groups defined by x).

If we take to be the i^{th} value of the k^{th} group and define the following values,

and and we assume that is normally distributed and are independent for all values of i and j with the expected normal values for the mean of

[26] See http://ccnmtl.columbia.edu/projects/qmss/the_ttest/twosample_ttest.html for detail.

and the standard deviation of , we can obtain the best unbiased linear estimate of and . These are defined to be:

is estimated by:

is estimated by:

The F test value (for the ANOVA F Test) is hence defined (Casella, 2002) as:

In this calculation, it is generally assumed that the population variance is equal (variance homeostasis or the homogeneity of variances). In this event, the value for can be written in a simplified form as:

This results in an F distribution with a central F variable with (K-1) and (N-K) degrees of freedom. This is of course one of the main failings with the ANOVA F test. If the sample variances differ, the simplified value of will no longer hold as being valid. As a consequence, the requirement for a suitable test of variances is necessary to ensure that homogeneity of variances exists.

Author Name, email@address

Tukey-Kramer HSD

Tukey-Kramer HSD (Honestly Significant Difference) test, or the Tukey–Kramer method, is a single-step multiple comparison procedure and statistical test generally used in conjunction with an ANOVA to find which means are significantly different from one another. The Tukey-Kramer HSD compares all possible pairs of means, and is based on a studentized range distribution q. The resultant distribution is comparable to the distribution of t from the t-test.

Tukey's test compares the means of every treatment to the means of every other treatment. Hence, the test can be seen to apply simultaneously to the set of all pairwise comparisons

.

This test identifies where the difference between two means is greater than the standard error would be expected to allow. The confidence coefficient for the set, when all sample sizes are equal, is exactly $1 - \alpha$. For unequal sample sizes, the confidence coefficient is greater than $1 - \alpha$. In other words, the Tukey method is conservative when there are unequal sample sizes.

What is p-value?

In a criminal trial, a suspect is assumed 'innocent' unless proven otherwise. In a statistical test, we commonly assume 'no difference' between two groups of data. The p-value is a number that is between 0 and 1, a value closer to 1 means there are 'no difference', a value closer to 0 means two groups of data are significantly different. Usually, the p-value needs to be less than 0.01 to establish there is significant difference between two groups of data. This cut-off value is chosen completely arbitrarily, and needs to be adjusted to 0.001 or less if higher stringency is required. Technically speaking, p-value of greater than 0.01 means the 'null hypothesis' is true, otherwise the 'alternative hypothesis' is true.

What is the alpha (α) value?

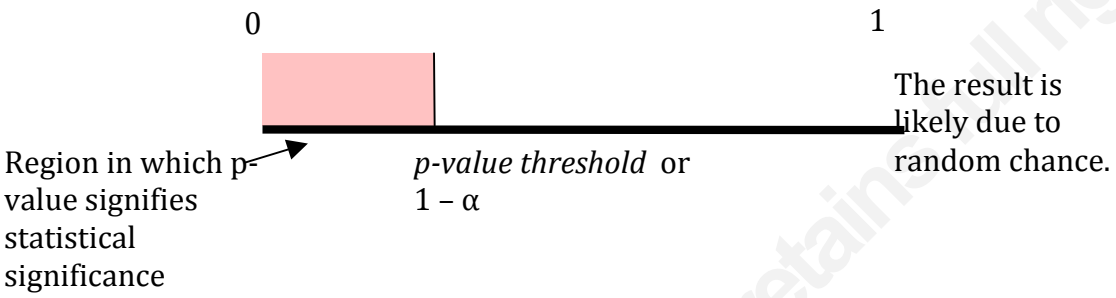
We have talked about the p-value threshold above. For a p-value cut-off of less than 0.05, the alpha value is 0.95, or more commonly expressed as a percentage 95%.

Author Name, email@address

This equation links α and p-value:

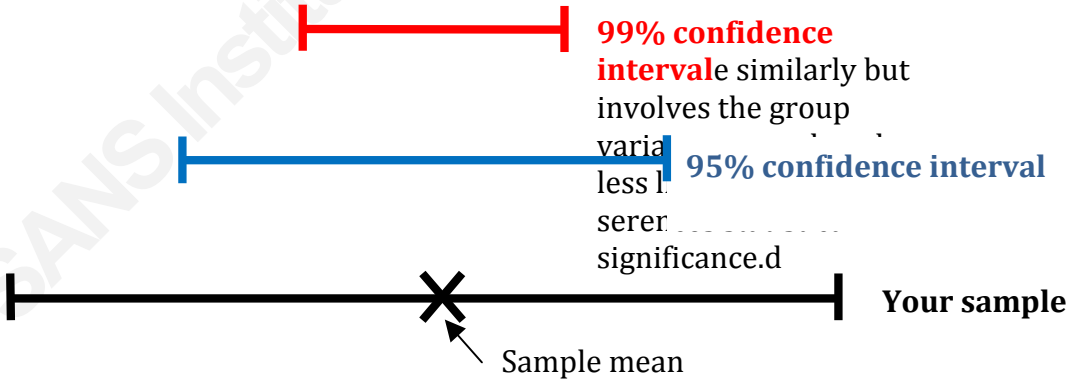
p-value threshold = 1 - α

By definition, the alpha value is the probability of incorrectly rejecting the null hypothesis.



What is confidence interval?

When you're considering one group of data, the confidence interval defines a range of values above and below the sample mean. A 95% confidence interval means that if we resample from the data, then it is expected that 95% of the data lies within the interval. A 99% confidence interval is always smaller than a 85% confidence interval.



The confidence interval is useful for the calculation of audit estimates within a particular α level. We can use old data to calculate a 95% confidence interval for a particular field in the spreadsheet. If the mean value for this year's data lies outside of the 95% confidence interval, then the result is statistically significant at p-value < 0.5. This also equals to a α value of 0.95. Since the value is different from previous year, this means this account may potentially have some problems that deserve to be further

Author Name, email@address

scrutinized. Therefore, confidence interval, alpha value, and p-value are closely related to each other.

What is correlation?

Correlation is a measure of how two sets of numbers move up or down together, and whether they have similar trends. One example of correlated data is the yearly unemployment rate and GDP of a country. In contrast, the number of birds that flew into my backyard each day and the Dow Jones figures for the day is likely to be uncorrelated. If the correlation value is 1, it means the data moves together up and down, if the correlation is -1, it means the data moves in separate ways. A correlation value of 0 means there are no relationship between the two sets of numbers.

5. HPing

Hping is THE tool for probing networks and injecting packets. For Pen testing with the crafting of an exploit packet, or in protocol fuzzing, HPing is an extremely versatile packet crafting tool.

You can download the latest version of hping source code from http://www.hping.org/

To install it, you need to compile it for your system. First change the file "libpcap_stuff.c" by modifying the line:

```
#include <net/bpf.h>
```

Change this to:

```
#include <pcap-net/bpf.h>
```

The process is a standard make from here:

#./configure

#make

#make install

HPing Operation

HPing will by default send TCP packets, with no options set, to port 0 of the target address continuously at one second intervals. The typical system response to this type of probe is an RST

packet, which indicates that a system is present. In addition to this default behavior, HPing offers the following command line options:

Hping3 [options] [target]

Protocol specific:

- --udp: generate UDP packets.
- --icmp: generate ICMP packets.
- --rawip: generate IP packets with no TCP or UDP components.

TCP Options:

- --syn: sets the SYN flag of a TCP packet.
- --fin: setis the FIN flag of a TCP packet.
- --rst: sets the RST flag of a TCP packet.
- --push: sets the PSH flag of a TCP packet.
- --ack: sets the ACK flag of a TCP packet.
- --urg: sets the URG flag of a TCP packet.

Network and port options:

- --rand-dest [IP-addr-net]: will send packets to random targets defined in the supplied network address block.
- --interface [int]: use the supplied interface for sending packets.
- --spoof [IP-addr]: use the supplied ip address as the 'spoofed' source address.
- --rand-source: randomize the source address.
- --destport [port]: supplied port will be used as the destination port. Adding + will increment the port by 1 for each received response, adding ++ will increment the port for each packet sent.
- --scan [port range/list]: supplied port range or list will be scanned.

- --baseport [port]: the supplied port will be used as the source port, which will increment by 1 for each packet sent. There is no + or ++ functionality. If no baseport is set, it is randomly selected (>1024).
- --keep: use a fixed source port for all packets

Speed options:

- --fast: send 10 packets per second (default is 1 packet per second).
- --faster: send 1,000,000 packets per second if possible
- --flood: send packets as fast as possible
- --interval [N], [uN]: send on packet every N seconds or every uN microseconds

Miscellaneous options:

- --count [N]: send N number of packets.
- --beep: beep when a packet is received.
- --file [filename]: use supplied file contents as payload. Requires –data option.
- --data [N]: length of payload to be sent in bytes.

The many different options which makes HPing a versatile tool for testing various aspects of systems or devices. Leveraging the random source address option (–rand-source), firewall rules can be tested for validity. The speed options can be used to test DoS mitigation, or detection sensitivity of IDS/IPS devices.

8 References

Bednarski, Greg M. and Branson, Jake; Carnegie Mellon University; "Information Warfare: Understanding Network Threats through Honeypot Deployment", March 2004

Broersma, Matthew; Techworld, "Linux servers safer than ever" Published 20th January 2005 by TechWorld http://www.techworld.com/security/news/index.cfm?NewsID=2983

CERT, "CERT/CC Statistics 1988-2005", 2005, http://www.cert.org/stats/cert_stats.html

CVE, CVE Site http://cve.mitre.org/data/downloads/allcves.html

Honeynet Project & Research Alliance, "Know Your Enemy: Honeywall CDROM Roo", 17th May 2005a, http://www.honeynet.org/papers/cdrom/roo/index.html

Honeynet Project & Research Alliance, "Know your Enemy: Tracking Botnets - Using honeynets to learn more about Bots", 13th March 2005b, http://www.honeynet.org/papers/bots/

Honeynet Project & Research Alliance, "Know Your Enemy: Honeynets in Universities - Deploying a Honeynet at an Academic Institution", 26th April 2004a, http://www.honeynet.org/papers/edu/

Honeynet Project & Research Alliance, "Know your Enemy: Trend Analysis", 17th December 2004b, http://www.honeynet.org/papers/trends/life-linux.pdf

Snort, Snort User Manual, www.snort.org/assets/166/snort_manual.pdf

Symantec, "Symantec Internet Security Threat Report", January 1 – June 30, 2004.

The SANS Institute, "Survival Time History", 2005, The Internet Storm Centre, http://isc.sans.org/survivalhistory.php

Wright, Craig S. & Zia, Tanveer, "*Compliance or Security, what cost? (Poster)",* ACISP 2011, 16th Australasian Conference on Information Security and Privacy, Melb. Au.