John Bruce Wallace

Internet Security: Intrusion, Spoofing, and SYN Floods

John Bruce Wallace Internet Security: Intrusion, Spoofing, and SYN Floods a paper on Web site management and security prepared for CMST 430 Web site Management, University of Maryland University College, December 6, 2006

TABLE OF CONTENTS

Internet Threats, Vulnerabilities, Security

With the geometric expansion of the prevalence of the Internet as the major global communications model has come the equal daunting specter of daily attacks against the informational data and assets of the global communications community. The temptation of potentially easy plunder of E-commerce sites along with traditional Web sites and corporate Intranets, through their LAN interfaces with the Internet, has driven cyber terrorism, cyber criminal activities, and of course the growth in awareness of the need for security. In a very basic sense all attacks against information assets are Intrusion attacks: an Intruder searches out the vulnerabilities of a computer, LAN, WAN, or Web site then exploits the discovered vulnerabilities for some form of gain. This paper will examine Intrusions in general, Web Spoofing, and a form of Denial of Service attack known as SYN Floods.

Intrusions: Attacks and Detection Systems

Intrusion:

An *intrusion* is an attack on information assets. According to Aurobindo Sundaram, as well as most other authorities, an Intrusion (I) is an unauthorized event where somebody attempts to break into or misuse your computer system generally with the intent to do malicious harm. An intrusion can be a simple annoyance that flashes non-sense on the computer monitor a single time or something as severe as stealing confidential data; destruction of the system; or misusing the system for spam, to mass virus attachments, or to implement a denial of service (DoS) attack. An "Intrusion Detection System (IDS)" is a system and policy procedures that are created and operated for detecting such intrusions. Per Sundaram, Intrusions can be divided into 6 main types:

- Attempted break-ins, which are detected by atypical behavior profiles or violations of security constraints.
- Masquerade attacks, which are detected by atypical behavior profiles or violations of security constraints.
- Penetration of the security control system, which are detected by monitoring for specific patterns of activity.
- Leakage, which is detected by atypical use of system resources.
- Denial of service, which is detected by atypical use of system resources.
- Malicious use, which is detected by atypical behavior profiles, violations of security constraints, or use of special privileges.

(Sundaram, 1996)

What Is A Typical Intrusion Scenario?

A typical intrusion scenario might include any or all of the following actions on the part of an intruder to gather information about the target computer or system that will facilitate the intruder's goal of misuse of the target:

- **Information Gathering**
  An attacker will normally start by finding out as much information as possible on

regarding the target, a process called *doorknob rattling*. At this point the attacker will want to be as stealthy as possible and will usually make use of less direct methods, processes called *footprinting* and *fingerprinting.* Some of these methods include doing a WHOIS lookup and DNS Zone transfers, as well as normal browsing of websites, gathering addresses, and similar important information belonging to the target.

- **Further Information Gathering**
  In an attempt to gather more information an attacker will usually perform ping sweeps, port scanning, and check Web servers for vulnerable CGI scripts. The intruder will also check the versions of running applications and services on the system's host - normally done using Banner Grabbing techniques. Typically banner grabbing consists of connecting to a service (for example SMTP on port 25) and parsing the response. In the response one would usually get the version of the application or a typical pattern of that application. A good IDS will catch some of this activity.
- **Attack!!**
  Having discovered usable information from the initial stealth invasion and now armed with a list of possible loopholes, the intruder will start trying out different attacks on the system. The intruder may, for example, try to launch a UNICODE attack if it was previously found out that the target has Microsoft Internet Information Services (IIS) installed. Apart from launching exploits for well known vulnerable software, a typical attacker will also try to find out any mis-configured services running on the target machine(s). A typical action that the intruder will try at this point is to guess passwords for known users on the system.
- **Successful intrusion**
  After a successful intrusion, attackers will usually install their own backdoors in the system and delete log files in order to hide their tracks. They may install 'toolkits' such as rootkits that establish alternative systems access, replace existing services with their own Trojan horses that have backdoor passwords, or create their own user accounts. System Integrity Checkers such as Tripwire have the task of detecting this kind of activity and alerting the administrator. From here an attacker will usually launch further attacks to other hosts, especially those that are trusted by the compromised machine.
- **Fun and profit**
  Different classes of system intruders have different goals. Some intruders steal confidential information such as credit card information, passwords to additional systems components, personal data such as bank account information, etc: while others just use the compromised host to launch further attacks on additional sites (such as DoS attacks). A few others will just deface a website. A recent, growing trend is to make use of a different pattern of attack. Intruders are increasingly randomly scanning internet addresses looking for a specific hole or number of holes, particularly on Windows/Intel machines which may be vulnerable due to the owner not having installed security patches. For example an intruder may scan for hosts having port 80 open and running a mis-configured/unpatched IIS server. Attackers will make a list of the vulnerable hosts and then launch attacks against each one of the hosts.

(iNFOSYSSEC)

The Need for Intrusion Detection Systems:

A computer system should provide *confidentiality*, *integrity,* and *assurance* against any intrusion. However, due to increased connectivity (especially on the Internet), and the vast spectrum of possibilities for financial gain that are opening up with more e-commerce sites being established, more and more systems are subject to attack by intruders. These subversion attempts try to exploit flaws in the operating system as well as in application programs. Such attacks have resulted in spectacular incidents like the Internet Worm incident of 1988[1].

There are two ways to handle subversion attempts. One way is to prevent subversion itself by building a completely secure system. Under such a system, for example, it would be a requirement that all users identify and authenticate themselves; data would be protected by various cryptographic methods in conjunction with very tight access control mechanisms. However this is not really feasible because:

- In practice, it is not possible to build a completely secure system. With regard to computer systems and software development, general industry knowledge regarding the issue of bugs in popular programs and operating systems seems to indicate that (a) bug free software is still a dream and (b) no-one seems to want to make the effort to try to develop such software. Apart from the fact that we do not seem to be getting our money's worth when we buy software, there are also security implications when our e-mail software, for example, can be attacked. Designing and implementing a totally secure system is thus an extremely difficult task.
- The vast number of installed base systems worldwide guarantees that any transition to a secure system, (if it is ever developed) will be long in coming.
- Cryptographic methods have their own problems. Passwords can be cracked, users can lose their passwords, and entire crypto-systems can be broken.
- Even a truly secure system is vulnerable to abuse by insiders who abuse their privileges.

[1] Spafford (n.d.). On the evening of 2 November 1988 the Internet came under attack from within. Some-time around 6 PM EST, a program was executed on one or more hosts connected to the Internet This program collected host, network, and user information, then broke into other machines using flaws present in those systems' software. After breaking in, the program would replicate itself and the replica would also attempt to infect other systems…The program was mysterious to users at sites where it appeared. Unusual files were left in the /usr/tmp directories of some machines, and strange messages appeared in the log files of some of the utilities, such as the *sendmail* mail handling agent. The most noticeable effect, however, was that systems became more and more loaded with running processes as they became repeatedly infected. As time went on, some of these machines became so loaded that they were unable to continue any processing; some machines failed completely when their swap space or process tables were exhausted

- It has been seen that that the relationship between the level of access control and user efficiency is an inverse one, which means that the stricter the mechanisms, the lower the efficiency becomes.

It is thus evident that under the prevailing state of technological abilities and industry values there is little alternative to the reality of systems that have vulnerabilities for a while to come. If there are attacks on a system, it is in an organization's best interests to detect them as soon as possible (preferably in real-time) and take appropriate action. This is essentially what an Intrusion Detection System (IDS) does. An IDS does not usually take preventive measures when an attack is detected; it is a reactive rather than pro-active agent. The IDS acts like a burglar alarm in that it detects a violation and signals an alarm. It plays the role of an informant rather than a police officer.

The root of all ID is based in analyzing a set of discrete sources, time-sequenced events for patterns of misuse. All ID sources are sequential records that directly reflect specific actions and indirectly reflect behavior. The most popular way to detect intrusions has been by using the *audit data* generated by the operating system. An *audit trail* is a record of activities on a system that are logged to a file in chronologically sorted order. Since almost all activities are logged on a system, it is possible that a manual inspection of these logs would allow intrusions to be detected. However, the incredibly large magnitude of audit data generated (on the order of 100 Megabytes a day) makes manual analysis impossible. IDSs automate the drudgery of wading through the audit data jungle. Audit trails are particularly useful because they can be used to establish guilt of attackers, should the organization pursue a criminal resolution or liability in a civil action, and they are often the only way to detect unauthorized but subversive user activity. (Sundaram, 1996; Proctor, 2001; Whitman, 2005)

Intrusion Detection:

Intrusion detection (ID) is the art of detecting and responding to computer misuse. It is the process of monitoring the events occurring in a computer system or network, and analyzing them for signs of *intrusions,* defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network. Intrusions are generally external attacks upon a system. Intrusions are caused by attackers attempting to access or successfully accessing an organizations information system(s) externally usually by way of the Internet, authorized users of the systems who attempt to gain additional privileges for which they are not authorized, and authorized users who misuse the privileges given them. Intrusion Detection Systems (IDSs) are software or hardware products that having been appropriately configured by an administrator automate this monitoring and analysis process.
(Bace, 2001; Proctor, 2001)

Intrusion Detection Systems:

Intrusion detection systems (IDSs) are software or hardware systems that automate the process of monitoring the events occurring in a computer system or network, analyzing them for signs of security problems. As network attacks have increased in number and severity over the past few years, intrusion detection systems have become a necessary addition to the security infrastructure of most organizations. While the software and hardware components of an IDS are important, perhaps the critical element in an organizations Intrusion Detection policy is the human factor. Given that this factor is notable for its inconsistency and instability, a highly defined IDP should be the corner-stone of an organization's security policy. It is essential for an organization to develop and implement through processes that include employee training an organizational culture that understands ID, its known causes and potential defenses. Such a policy includes the need to identify those who need to understand what security goals ID mechanisms serve, how to select and configure ID systems for their specific system and network environments, how to manage the output of ID systems, and how to integrate ID functions with the rest of the organizational security infrastructure.

Intrusion detection allows organizations to protect their systems from the threats that come with increased network connectivity and reliance on external information systems that are necessary for communication, corporate business, and e-commerce transactions. The necessity of interaction between an organization's information system and the unknowns that may be present on the global frontier means that every transaction contains extra risk. Given the level and nature of modern network security threats, the question for security professionals should not be *whether* to use intrusion detection, but *which* intrusion detection features and capabilities to use. IDSs have gained acceptance as a necessary addition to every organization's security infrastructure. Despite the documented contributions ID technologies make to system security, in many organizations one must still justify the acquisition of IDSs.
(Bace, 2001; Proctor, 2001)

Generally speaking, there are five different categories of intrusion detection systems: network intrusion detection systems, host-based intrusion detection systems, system integrity verifiers, log file monitors, and deception systems, although the last three are sometimes seen as versions of application-based systems, which in turn are sometimes viewed as host-based variations.

- Network intrusion detection systems (NIDS) monitor packets traversing the system in an attempt to discover anomalies, indicating that an intruder may be trying to break into a system, or worse — possible launch a distributed denial of service (DDoS) attack. NIDSs look for frequent connection requests to different ports to reveal port scans.
- Host-based intrusion detection systems (HIDS) reside on a specific host computer and monitor the activity of the host machine by monitoring security event logs or checking for changes to the system, for example changes to critical system files or to the systems registry.
- System integrity verifiers (SIV) monitor system files in an attempt to discover when an intruder changes the files — leaving behind a backdoor. A SIV may be capable of detecting changes in critical files, but these systems usually don't generate real-time alerts to network intruders.
- Log file monitors (LFM) simply monitor log files generated across network services. LFMs also look for patterns and anomalies in log files that suggest an intruder is attacking the network.

- The sole purpose of a deception system—known in the industry as decoys, fly traps and honeypots—is to lure an unsuspecting intruder into a network through well-known security holes and trap the intruder. However, the use of these systems should be used with extreme care as their use may carry a greater risk of liability for the organization in that by entrapping an intruder or worse by conducting a reverse intrusion (back-hacking) into the intruder's system the corporation may be equally guilty of breaking the law and subject to the resulting criminal complaint and penalties.

(Jupitermedia, 2006; Bace, 2001; iNFOSYSSEC; Proctor, 2001; Sundaram, 1996; Whitman, 2005)

Intrusion Response:

Intrusion management needs significant administrative attention and effort to be efficient. The core to a sound Intrusion Detection Policy (IDP) is the development of a comprehensive IDP. Intrusion detection by itself does not mitigate risks of an intrusion. Risk mitigation only occurs through an effective and timely response. The goal of the response is to minimize damage to the institution and its customers through containment of the intrusion, and restoration of systems.

The response primarily involves people rather then technologies. The quality of intrusion response is a function of the institution's culture, policies and procedures, and training. Preparation determines the success of any intrusion response. Preparation involves defining the policies and procedures that guide the response, assigning responsibilities to individuals and providing appropriate training, formalizing information flows, and selecting, installing, and understanding the tools used in the response effort. Successful implementation of any response policy and procedure requires the assignment of responsibilities and training.
(FFIEC)

Intrusion Conclusions:

Intrusion Detection is still a fledgling field of research. However, it is beginning to assume enormous importance in today's computing environment. The combination of facts such as the unbridled growth of the Internet, the vast financial possibilities opening up in electronic trade, and the lack of truly secure systems make it an important and pertinent field of research. Future research trends seem to be converging towards a model that is a hybrid of the current detection models; it is slowly acknowledged that none of the models can detect all intrusion attempts on their own. This approach has been successfully adopted in NIDES, and we can expect more such attempts in the future.

However, reliance on technology to provide solutions to what are in effect behavior traits is to be short-sighted. As previously indicated, IDSs can only alert to the possibility of an intrusion or misuse. The determination must be made by someone in an administrative capacity. A human must read and interpret the characteristics precipitous of any attack perpetrated by another human. Reflection on the following example provides useful insight into the degree to which human interaction is an underlying factor. Consider the case where an organization's workstations are found to be performing intense activity at an unusual time, when under normal circumstances the workstations would not be in use, it could be that upper management has arranged with the local university for the workstations to be utilized as part of a super-computer to conduct an intense and significantly large research project and neglected to inform Security and the Information Departments, or it could be that an intruder has arranged without management approval for the workstations to take part in a DoS on some government organization. The determination as to whether to declare that an intrusion has occurred is a human decision dependent on all the attending factors. A comprehensive IDP that has pervaded the organizational culture will make the job of making that determination much easier. In fact the ability of the IDS to alert to the potential of an intrusion is dependent on the ability of the systems administrator and programmers to prepare adequate priorities configured into the IDS that will facilitate the recognition of the potential intrusion. This seems to make the human factor the significant element in any IDS.

Web Spoofing Attack Scenario[2]

Scenario:

Web Spoofing is a collection of security attacks that allow an adversary to observe and modify all web pages requested by or sent from the computer or system that is the object of the attack (victim), as-well-as observe all information entered into forms by the victim. Web Spoofing works on both of the major browsers and is *not* prevented by "secure" connections. During an attack the attacker gains access to the target system or computer and can observe and modify all requested and transmitted web pages and form submissions from the target system or computer, even when the browser's "secure connection" indicator is lit. During a spoofing attack from the victim's perspective there is no indication that anything is amiss.

In a *spoofing attack*, the attacker creates a misleading context (false Web reality) in order to trick the victim into making an inappropriate security-relevant decision. Analogous to the methods employed in a con game: during a spoofing attack the attacker sets up a false but convincing world around the victim. Unaware that the system or computer is subject to a successful attack and penetration the victim proceeds to perform whatever real world tasks would be appropriate if the attacker modified false Web environment were real. Unfortunately, reasonable real Web world activities, procedures, and practices that a victim may employ in the false world may have disastrous real world effects and consequences.

What the Attacker Does:

Web spoofing facilitates a virtually altered cyberspace in which the attacker creates a convincing but false copy of the entire World Wide Web, a "shadow copy" of the entire Web. Accesses to the shadow Web are funneled through the attacker's server, allowing the attacker to monitor all of the victim's activities including any passwords or account numbers the victim enters. The attacker can also cause false or misleading data to be sent to Web servers in the victim's name, or to the victim in the name of any Web server, so that all network traffic between the victim's browser and the Web goes through the attacker. In short, the nature of the attack permits the attacker to observe and control everything the victim does on the Web.

Since the attacker can observe or modify any data going from the victim to Web servers, as well as controlling all return traffic from Web servers to the victim, the attacker has many possibilities, including surveillance and tampering.

- Surveillance The attacker can passively watch the traffic, recording which pages the victim visits and the contents of those pages. When the victim fills out a form, the entered data is transmitted to a Web server. Since this data travels through the altered route that passes through the attacker's system the attacker can record that the entire bi-directional transaction, data sent along with the response sent back by the server. Since most on-line commerce is done via forms, this means the attacker can observe *any account numbers or passwords the victim enters*.

[2] This scenario as discussed through the Section Header "Why the Attack Works" has been adapted from Felten, 1997

The attacker can carry out surveillance even if the victim has a “secure” connection (usually via Secure Sockets Layer) to the server, that is, even if the victim’s browser shows the secure-connection icon (usually an image of a lock or a key).

- Tampering   The attacker is also free to modify any of the data traveling in either direction between the victim and the Web. The attacker can modify any and all form data submitted by the victim. For example, if the victim is ordering a product on-line, the attacker can change the product number, the quantity, or the ship-to address.
  The attacker can also modify all data returned by a Web server. A possible scenario would find the attacker inserting misleading or offensive material into a form or Web page in order to trick the victim or to cause antagonism between the victim and the server.

During a Spoofing attack, since the whole Web is available on-line, the attacker need only fetch a page from the real Web when needed to provide a copy of that page for use on the false Web.

Triggering the Attack:

The attack is initiated when the victim visits a malicious Web page, or receives a malicious email message (if the victim uses an HTML-enabled email reader). To start an attack, the attacker must somehow lure the victim into the attacker’s false Web. There are several ways to do this including but not limited to:

- An attacker could put a link to a false Web onto a popular Web page.
- If the victim is using Web-enabled email, the attacker could email the victim a pointer to a false Web, or even the contents of a page in a false Web.
- Finally, the attacker could trick a Web search engine into indexing part of a false Web.

The attack is normally implemented using JavaScript and Web server plug-ins, and works in two parts.

- First, the attacker causes a browser window to be created on the victim's machine, with some of the normal status and menu information replaced by identical-looking components supplied by the attacker.
- Then, the attacker causes all Web pages destined for the victim's machine to be routed through the attacker's server. On the attacker's server, the pages are rewritten in such a way that their appearance does not change at all, but any actions taken by the victim (such as clicking on a link) would be logged by the attacker. Any attempt by the victim to load a new page would cause the requested new page to be loaded by routing through the attacker's server, so the attack would continue on the new page.

Why the Attack Works:

The key to this attack is for the attacker’s Web server to sit between the victim and the rest of the Web. This kind of arrangement is called a “man in the middle attack” in the security literature. The attack is primarily accomplished by the attacker employing a revision of the real Web URL to reflect the URL of the Shadow Web.

URL Rewriting

The attacker's first trick is to rewrite all of the URLs on some Web page (the lure) so that they point to the attacker's server rather than to some real server. Assuming the attacker's server is on the machine `www.attacker.org`, the attacker rewrites a URL by adding `http://www.attacker.org` to the front of the URL. For example, `http://home.netscape.com` becomes `http://www.attacker.org/http://home.netscape.com`

When the victim requests a page through one of the rewritten URLs, the victim's browser requests the page from `www.attacker.org`, since the URL starts with `http://www.attacker.org`. The remainder of the URL tells the attacker's server where on the Web to go to get the real document.

The attacker's server retrieves the real document needed to satisfy the request, whereupon the attacker rewrites all of the URLs in the document into the same special form by splicing `http://www.attacker.org/` onto the front portion of the URL. Then the attacker's server provides the rewritten page to the victim's browser.

Since all of the URLs in the rewritten page now point to `www.attacker.org`, if the victim follows a link on the new compromised page, any pages and materials at the requested link will again be retrieved through the attacker's server. The victim remains trapped in the attacker's false Web, and can follow links forever without leaving it, trapped on a virtual Mobius Strip.

If the victim fills out a form on a page in a false Web, the result appears to be handled properly. Spoofing of forms works naturally because forms are integrated closely into the basic Web protocols, form submissions are encoded in Web requests and the replies are ordinary HTML. Since any URL can be spoofed, forms can also be spoofed.

Once an attacker has compromised a system or computer, when the victim submits a form, the submitted data goes to the attacker's server. The attacker is able therefore to observe and even modify the submitted data, doing whatever malicious editing desired, before passing it on to the real server. The attacker can also modify the data returned in response to the form submission as it passes through the attacker's server.

An example of a Web transaction executed during a successfully established Web spoofing attack. Upon the victim's request of a Web page the following steps occur:

(1) the victim's browser requests the page from the attacker's server;
(2) the attacker's server requests the page from the real server;
(3) the real server provides the page to the attacker's server;
(4) the attacker's server rewrites the page;

(5) the attacker's server provides the rewritten version to the victim

Remedies:

Web spoofing is a dangerous and nearly undetectable security attack that is regularly carried out on today's Internet. Although an attack cannot be completely prevented there are some protective measures that can be taken to minimum the effects of an attack. Most of the Anti-Spoofing measures shall be implemented by the IT Department, unless otherwise noted.

Pre-attack defenses:

Short-term Solution:

In the short run, the best defense is to follow a three-part strategy:

1. where possible disable JavaScript, ActiveX, and Java in the browser so the attacker will be unable to hide the evidence of the attack;
2. make sure the browser's location line is always visible;
3. End users must pay attention to the URLs displayed on the browser's location line, noting any irregularities.

Long-term Solution:

There is currently no fully satisfactory long-term solution to this problem. Changing browsers so they always display the location line would help, although users would still have to be vigilant and know how to recognize rewritten URLs. This is an example of a "trusted path" technique, in the sense that the browser is able to display information for the user without possible interference by un-trusted parties. For pages that are not fetched via a secure connection, there is not much more that can be done. For pages fetched via a secure connection, an improved secure-connection indicator could help. Rather than simply indicating a secure connection, browsers should clearly say who is at the other end of the connection.
(Felten, 1997)

With the current IP protocol technology, it is impossible to eliminate IP-spoofed packets. However, a system administrator can take steps to reduce the number of IP-spoofed packets entering and exiting a network.

Currently, the best method is to install a filtering router that restricts the input to the external interface (known as an input filter) by not allowing a packet through if it has a source address from the internal network. In addition, system administrators should filter outgoing packets that have a source address different from the internal network to prevent a source IP spoofing attack from originating from an organization's site.

The combination of these two filters would prevent outside attackers from sending packets pretending to be from the internal network. It would also prevent packets originating within the network from pretending to be from outside the network. These filters will not stop all TCP SYN attacks, since outside attackers can spoof packets from any outside network, and internal attackers can still send attacks spoofing internal addresses.

If the routers do not support filtering on the inbound side of the interface or if there will be a delay in incorporating the feature into the system, a system administrator may filter the spoofed IP packets by using a second router between the external interface and the outside connection. Configure this router to block, on the outgoing interface connected to the original router, all packets that have a source address in the internal network. For this purpose, use a filtering router or a UNIX system with two interfaces that supports packet filtering.

Note: Disabling source routing at the router does not protect a system from this attack, but it is still good security practice to follow.

On the input to the external interface, coming from the Internet to the internal network, a system administrator should block packets with the following addresses:

- Broadcast Networks: The addresses to block here are network 0 (the all zeros broadcast address) and network 255.255.255.255 (the all ones broadcast network).
- The local network(s): Block the internal network addresses
- Reserved private network numbers: The following networks are defined as reserved private networks, and no traffic should ever be received from or transmitted to these networks through a router:
    - 10.0.0.0 - 10.255.255.255 10/8 (reserved)
    - 127.0.0.0 - 127.255.255.255 127/8 (loopback)
    - 172.16.0.0 - 172.31.255.255 172.16/12 (reserved)
    - 192.168.0.0 - 192.168.255.255 192.168/16 (reserved)

(CERT, 2000; Killalea, 2000)

Spoofing Conclusions:

As with Intrusions, the best defense against spoofing is an offense where the end user is on constant vigilance lookout for the unexpected, figurative and literally. Such a heightened awareness is fostered by a sound and viable security awareness, training, and education program aimed at making security the organizations mantra.

SYN Flood Attack Scenario[3]

Denial of Service Attacks:

Denial-of-service attacks continue to plague Internet sites. Denial-of-service (DoS) is a class of attacks in which an attacker attempts to prevent legitimate users from accessing an Internet service, such as a web site. This can be done by exercising a software defect that causes the software running the service to fail, an example of a DoS is the “Ping of Death” attack against Windows NT systems, sending enough data to consume all available network bandwidth or sending data in such a way as to consume a particular resource needed by the service. One type of denial-of-service attack, the SYN flood, is particularly effective at disabling Web servers.

SYN Flood Scenario:

TCP SYN Flooding also known as “Resource-exhaustion attacks” causes servers to quit responding to client requests to open new connections -- a denial of service attack. Named for the SYN packet that initiates a TCP/IP connection, a SYN flood consumes all available slots in a server’s TCP connections table, and by doing so, prevents other users from establishing new TCP/IP connections. Hypertext Transfer Protocol (HTTP), the predominate protocol for the World Wide Web, is particularly vulnerable to a SYN flood attack. As a matter of course in the process of obtaining a HTTP Web page, Web browsers establish one or more new TCP/IP connections to the web server for every web page viewed, so if new connections cannot be established, the web server is essentially unusable.

How SYN Floods Work:

A SYN flood exploits a basic weakness in the TCP/IP protocol. The SYN in SYN flood stands for the Synchronize flag in TCP data-packet headers. As part of the Open System Interconnection Model (OSI) communication process establishing a new TCP/IP network connection requires a three-step process[4]:

---

[3] This scenario adapted from Mavens, 2001 and Schetina, 2002.

[4] An example of the three-step handshake that establishes an Internet session, as examined in the packet sniffer WireShark (f/n/a Ethereal) follows:

```
No.  Time        Source           Destination      Protocol  Info
79 16.844496     128.192.98.130   128.192.98.53    DNS       Standard query A www1.anyorg.com
80 16.864484     128.192.98.53    128.192.98.130   DNS       Standard query response CNAME www1.anyorg.com A
207.217.125.95
81 16.865129     128.192.98.130   www1.anyorg.com  TCP       1189 > http [SYN] Seq=0 Ack=0 Win=65535 Len=0 MSS=1460
88 16.957572     128.192.98.130   www1.anyorg.com  HTTP      GET / HTTP/1.1
89 17.089254     www1.anyorg.com  128.192.98.130   TCP       http > 1189 [ACK] Seq=1 Ack=425 Win=48856 Len=0
90 17.097432     www1.anyorg.com  128.192.98.130   HTTP      HTTP/1.1 301 Moved Permanently (text/html)
91 17.100851     128.192.98.130   www1.anyorg.com  TCP       1190 > http [SYN] Seq=0 Ack=0 Win=65535 Len=0 MSS=1460
92 17.191714     www1.anyorg.com  128.192.98.130   TCP       http > 1190 [SYN, ACK] Seq=0 Ack=1 Win=8190 Len=0 MSS=1408
93 17.191812     128.192.98.130   www1.anyorg.com  TCP       1190 > http [ACK] Seq=1 Ack=1 Win=65535 Len=0
94 17.192872     128.192.98.130   www1.anyorg.com  HTTP      GET / HTTP/1.1
95 17.224054     128.192.98.130   www1.anyorg.com  TCP       1189 > http [ACK] Seq=425 Ack=519 Win=65017 Len=0
96 17.324222     www1.anyorg.com  128.192.98.130   TCP       http > 1190 [ACK] Seq=1 Ack=636 Win=49280 Len=0
97 17.343942     www1.anyorg.com  128.192.98.130   TCP       [TCP segment of a reassembled PDU]
98 17.358932     www1.anyorg.com  128.192.98.130   TCP       [TCP segment of a reassembled PDU]
99 17.358978     128.192.98.130   www1.anyorg.com  TCP       1190 > http [ACK] Seq=636 Ack=2817 Win=65535 Len=0
```

1. The originator of the connection initiates the connection by sending a packet having the SYN flag set in the TCP header (referred to as a "SYN packet") down the OSI stack.
2. The receiver responds by sending back to the originator a packet that has the SYN and ACK (Acknowledgement) flags set (a "SYN/ACK packet").
3. The originator acknowledges receipt of the 2nd packet by sending to the receiver a third packet with only the ACK flag set (an "ACK packet").

Only after this three-step handshaking has been completed is the TCP connection considered "open" allowing data exchange to begin between the originator and recipient. Between steps 2 and 3, the receiver computer must keep a record of this incomplete connection while waiting for the ACK packet.

The TCP SYN Flooding attacker takes advantage of this aspect of how the TCP protocol establishes a new connection. During a SYN flood attack, the attacker sends a large number of SYN packets alone, without the corresponding ACK packet response to the victim's SYN/ACK packets. Most systems have only a limited amount of space for such records. Since the information stored takes up memory and operating system resources, only a limited number of in-progress connections are allowed. The victim's connections table rapidly fills with incomplete connections consuming all available TCP connectivity resources and consequently crowding out the legitimate traffic. Most TCP/IP implementations impose a relatively long timeout period (several minutes) before incomplete connections are cleared out leaving a large window of opportunity for an intruder (attacker) to launch a SYN flood attack. Because the rate of attacking SYN packets usually far exceeds that of normal traffic, even when a table entry eventually is cleared out, another attacking SYN packet rather than a legitimate connection will fill it. Software on the attacker's computer or system generates false packets that appear to be valid new connections to the server. These fictitious packets enter the queue, but the connection is never completed -- leaving these new connections in the queue until they time out. Only a few of these packets need to be sent to the server, making this attack simple to carry out even using a

```
100 17.373951   www1.anyorg.com   128.192.98.130    TCP    [TCP segment of a reassembled PDU]
101 17.451943   128.192.98.130    www1.anyorg.com   TCP    1191 > http [SYN] Seq=0 Ack=0 Win=65535 Len=0 MSS=1460
102 17.466601   www1.anyorg.com   128.192.98.130    TCP    [TCP segment of a reassembled PDU]
103 17.466695   128.192.98.130    www1.anyorg.com   TCP    1190 > http [ACK] Seq=636 Ack=5633 Win=65535 Len=0
104 17.481794   www1.anyorg.com   128.192.98.130    TCP    [TCP segment of a reassembled PDU]
105 17.496470   www1.anyorg.com   128.192.98.130    TCP    [TCP segment of a reassembled PDU]
106 17.496572   128.192.98.130    www1.anyorg.com   TCP    1190 > http [ACK] Seq=636 Ack=8449 Win=65535 Len=0
107 17.544299   www1.anyorg.com   128.192.98.130    TCP    http > 1191 [SYN, ACK] Seq=0 Ack=1 Win=8190 Len=0 MSS=1408
108 17.544392   128.192.98.130    www1.anyorg.com   TCP    1191 > http [ACK] Seq=1 Ack=1 Win=65535 Len=0
109 17.545325   128.192.98.130    www1.anyorg.com   HTTP   GET /scripts/common.css HTTP/1.1
110 17.574094   www1.anyorg.com   128.192.98.130    TCP    [TCP segment of a reassembled PDU]
111 17.589026   www1.anyorg.com   128.192.98.130    TCP    [TCP segment of a reassembled PDU]
112 17.589142   128.192.98.130    www1.anyorg.com   TCP    1190 > http [ACK] Seq=636 Ack=11265 Win=65535 Len=0
113 17.604160   www1.anyorg.com   128.192.98.130    TCP    [TCP segment of a reassembled PDU]
114 17.604441   128.192.98.130    www1.anyorg.com   TCP    1190 > http [ACK] Seq=636 Ack=12673 Win=65535 Len=0
115 17.621737   www1.anyorg.com   128.192.98.130    TCP    [TCP segment of a reassembled PDU]
116 17.636657   www1.anyorg.com   128.192.98.130    TCP    [TCP segment of a reassembled PDU]
117 17.636695   128.192.98.130    www1.anyorg.com   TCP    1190 > http [ACK] Seq=636 Ack=15489 Win=65535 Len=0
118 17.651842   www1.anyorg.com   128.192.98.130    TCP    [TCP segment of a reassembled PDU]
119 17.651896   128.192.98.130    www1.anyorg.com   TCP    1190 > http [ACK] Seq=636 Ack=16897 Win=65535 Len=0
120 17.671867   www1.anyorg.com   128.192.98.130    TCP    http > 1191 [ACK] Seq=1 Ack=510 Win=48771 Len=0
121 17.674609   www1.anyorg.com   128.192.98.130    HTTP   HTTP/1.1 304 Not Modified
122 17.675788   128.192.98.130    www1.anyorg.com   HTTP   GET /scripts/common.js HTTP/1.1
```

slow, dial-up (like PPP or SLIP) connection from the attacker's computer. The system under attack quits responding to new connections shutting down until some time after the attack stops. (Farrow, 2006; Mavens, 2001; Schetina, 2002)

Although a form of Denial-of-service attack and bearing many of the hallmarks of such attacks, SYN flood attacks are not necessarily easy to defend against. Three characteristics make SYN flood attacks difficult to counter.

- First, data packets that cause other types of denial-of-service attacks, such as “Ping of Death” or “Smurf”, can be blocked entirely by a firewall. But because SYN packets are a necessary part of legitimate traffic, they cannot be filtered out altogether.
- Second, SYN packets are relatively small, so an attacker can send large numbers of packets using relatively low-bandwidth Internet connections.
- Finally, because the attacker does not need to receive any data from the victim, the attacker can employ spoofing techniques placing random source IP addresses in the attacking packets to camouflage the actual source of the attack, and make filtering all but impossible.

(Mavens, 2001 and Schetina, 2002)

Responding to SYN Floods:

Given the system components involved during a SYN flood, increasing the size of the network connections table would seem to be the most straightforward way to mitigate the effect of a SYN flood. However, in most operating systems the table is normally part of the operating system kernel, and may not be configurable. Even if the table can be resized, it would need to be enlarged considerably to be effective against a SYN flood of more than a few seconds. Such a large table would significantly degrade network performance, or require major redesign of the network code within the operating system kernel. Suggested solutions have also considered employing “hot spare” servers to be placed in service during an attack. This is essentially an attempt to “out horsepower” the attacker. However, having expensive server equipment sitting idle until an attack occurs is a very expensive option, not likely to be economically viable. A second negative of this proposed solution is that it would take vital time to respond with the “hot spare”; time that would include the period during the verification that an attack was in fact real, time to boot the hot spare and bring it online. During this critical time window an attack could do a lot of damage.

Identifying the origin of the attack is another approach. But if the attacker uses spoofed source IP addresses, locating the source would involve tracing the actual network traffic through routers and ISPs, a very time-consuming process. In the case of a distributed attack, identifying all compromised machines and contacting their respective owners could take weeks or even months, an exceptionally costly and inefficient solution to an immediate incident requiring an immediate response.

Furthermore in many instances susceptibility to SYN flood attacks is a direct consequence of trusting other stakeholders and systems to play fairly with a sense of honor and proper Internet decorum. The attacker knows the rules and intentionally does not play by them.

Importantly the flaw in the system can also be allocated to the lack of adequate developers trained to incorporate complex security algorithms into the vulnerable system or application. (Farrow, 2006; Mavens, 2001; Schetina, 2002)

Remedies:

Although some have described TCP SYN Flooding as a ``bug" in TCP/IP, it is more correctly a ``feature" of the design. TCP/IP was designed for a friendly Internet where participating parties play by the rules, as has been noted, and a limited connection queue has worked fine for years.

Early fixes have focused on increasing the length of the queues and reducing a timeout value. The timeout value controls how long an entry waits in the queue until an acknowledgement is received. The problem with simply making the queue longer is that there are actually many queues (one for each TCP server on the system--HTTP, FTP, SMTP, etc.), and lengthening the queues to very large values results in an operating system requiring enormous amounts of memory.

Shortening the timeouts can also help when used with longer queue lengths because the spoofed packets get removed from the queues more quickly. Shortening the timeouts also affects new outgoing connections and remote users with slow links increasing the connectivity opportunities to those users

The best solution is to modify the TCP implementation to reduce the amount of information stored for each in-progress connection. Another modification checks the return route of the new connection, and if it is different than the received packets (which is normal during this attack) to drop this connection.
(Farrow, 2006)

To provide the ability to endure SYN flood attacks, at least temporarily, manufacturers of firewalls and other network security devices have developed a variety of defense methods to incorporate into their products. Some of the methods currently on the market include:

- SYN threshold: establishes a limit or quota on the number of incomplete connections, and discards SYN packets if the number of incomplete connections reaches the limit.
- SYNDefender: when a SYN packet is received, the firewall synthesizes the final ACK packet implemented during the third step in the TCP/IP handshake so the receiver does not need to wait for the actual ACK packet from the originator.
- SYN proxy: the firewall synthesizes and sends the SYN/ACK packet back to the originator, and waits for the final ACK packet. After the firewall receives the ACK packet from the originator, the firewall then “replays” the three-step sequence to the receiver.
- SYN cookies: this method attempts to eliminate the need to store incomplete connection information by including a package of information, or a “cookie” in the SYN/ACK packet sent by the receiver to the originator. When the originator responds with the ACK

packet, the cookie is returned and the receiver is able to extract the information needed to rebuild the connection.

(Farrow, 2006; Mavens, 2001; Schetina, 2002)

SYN Flood Conclusions:

While SYN floods continue to be one of the most challenging types of network attacks to defend against, effective defenses as indicated are available in the marketplace today although there is a wide variation in effectiveness of the different products on the market showing the critical importance of verifying performance before relying on any particular product or technique as a defense to a SYN flood attack. Another aspect to consider when selecting a device is ease of deployment. Many security incidents catch organizations unprepared, and speed of response to an attack is very important.

Final Observations:

Intrusion management needs significant administrative attention and effort to be efficient. The core to a sound Incident and Disaster Response and Recovery Policy (IDRRP) is the development of a comprehensive organizational awareness of the stakes at issue. Intrusion detection by itself does not mitigate risks of an intrusion. Risk mitigation only occurs through an effective and timely response. The goal of the response is to minimize damage to the institution and its customers through containment of the intrusion, and restoration of systems.

While defense against external attacks is definitely necessary, the growth in distributed attacks shows that preventing an organization's systems from being unwilling participants in an attack is equally important. Here again the importance of implementation of an effective defensive network that includes ID systems, a Demilitarized Zone, firewalls, ingress and egress scripted configurations of router filters, and of course a strong security policy that has top management support for an organizational awareness of security and its central place within the organizational culture cannot be overstated. Prevention, to the extent possible through a comprehensive security awareness, training, and education program, while not the panache solution will go a long way in establishing an organizational environment where security is not a stranger. Still in the end these attacks are the conception of individuals that have chosen to not play by the rules leaving vulnerabilities at the doorstep of exploitation.

References:

Andrews, J., and Beck, W. (2005). *i-Net+ Guide to the Internet*, 3'd Ed. Boston, MA: Thomson

Andrews, M, and Whittaker, J. (2006). *How to Break Web Software*. Upper Saddle River, NJ: Addison-Wesley

Bace, R. and Mell, P. (2001, August 16). *Intrusion Detection Systems*. National Institute of Standards and Technology (NIST), Special Publication 800-31. Retrieved June 22, 2006, from http://csrc.nist.gov/publications/nistpubs/800-31/sp800-31.pdf

CERT (2000/1996). CERT® Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks. Pittsburgh PA: CERT Coordination Center Software Engineering Institute, Carnegie Mellon University. Retrieved November 16, 2006, from http://www.cert.org/advisories/CA 1996-21.html

Cisco (2005, December 20). Cisco Security Response: Response to VLAN and PVLAN Bidirectional Jumping. Cisco Systems Product Security Incident Response Team. Retrieved November 17, 2006 from http://archives.neohapsis.com/archives/cisco/2005-q4/0013.html

Dr-K. (2000). *A Complete H@CKER'S Handbook Everything You need to Know About Hacking in the Age of the Web*. London, England: Carlton Books

Durst, R., Champion, T., Witten, B., Miller, E., and Spagnuolo, L. (1999, July). Testing and Evaluating Computer Intrusion Detection Systems. *Communications of the ACM*, v42 i7 p53. Retrieved November 26, 2006 from http://web5.infotrac.galegroup.com.ezproxy.umuc.edu/itw/infomark/790/396/94991483w5/purl=rc1_CDB_0_A55198495&dyn=3!xrn_16_0_A55198495&bkm_3_13?sw_aep=umd_umuc

Farrow, R. (2006). TCP SYN Flooding Attacks and Remedies. Networking Computer for IT. Retrieved November 21, 2006 from http://www.networkcomputing.com/unixworld/security/004/004.txt.html

Federal Financial Institutions Examination Council (FFIEC). (n.d.). Intrusion Detection and Response. Security Controls Implementation. Information Security Booklet. Retrieved July 4, 2006, from http://www.ffiec.gov/ffiecinfobase/booklets /information_security/04j_intrusion_detect%20_response.htm

Felten, E., Balfanz, D., Dean, D. and Wallach, D. (1997). *Web Spoofing: An Internet Con Game*, Technical Report 540–96. Department of Computer Science, Princeton University. Retrieved November 13, 2006 from http://www.cs.princeton.edu/sip/pub/spoofing.doc

Jupitermedia Corporation. (2006). ISP-Planet Intrusion Detection Systems Directory. Retrieved July 3, 2006, from http://www.isp-planet.com/services/ids/index.html

iNFOSYSSEC (n.d.). The Security Portal for Information Security Professionals. Intrusion Detection and Network Auditing on the Internet - Articles, Tutorials, Surveys. Retrieved June 19, 2006, from http://www.infosyssec.net/infosyssec/security/intdet1.htm

Kairab, S. (2005). *A Practical Guide to Security Assessments*. Boca Raton, FL: Auerbach Publications

Killalea, T., (2000). Recommended Internet Service Provider Security Services and Procedures. The Internet Society. Retrieved November 16, 2006 from http://rfc.net/rfc3013.html#s4.1

Mandia, K., Prosise, C., and Pepe, M. (2003). *Incident Response & Computer Forensics,* 2'd Ed. New York, NY: McGraw-Hill

Martz, C. (n.d.). Detection System. Retrieved July 3, 2006, from http://www.birds-eye.net/definition/n/nids-network_intrusion_detection_system.shtml

Mavens, T. (2001, August 29). Countering SYN Flood Denial-of-Service Attacks. Tech Mavens Retrieved November 21, 2006 from http://www.tech-mavens.com/synflood.htm

Mell, P., Hu, V., Lippmann, R., Haines, J., and Zissman, M. (n.d.). An Overview of Issues in Testing Intrusion Detection Systems. National Institute of Standards and Technology. Retrieved July 4, 2006, from http://citeseer.ist.psu.edu/cache/papers/cs/29833/http:zSzzSzcsrc.nist.govzSzpublicationszSznistirzSznistir-7007.pdf/an-overview-of-issues.pdf

Newmann, P. (2000, April). Denial-of-Service Attacks. *Communications of the ACM*, v43 i4 p136. Retrieved November 26, 2006 from http://web5.infotrac.galegroup.com.ezproxy.umuc.edu/itw/infomark/790/396/94991483w5/purl=rc1_CDB_0_A61792764&dyn=3!xrn_13_0_A61792764&bkm_3_13?sw_aep=umd_umuc

Patterson, T. (2005). *Mapping Security*. Upper Saddle River, NJ: Addison Wesley

Porras, P. (2006). Next-Generation Intrusion Detection Expert System (NIDES). SRI International. Retrieved July 4, 2006, from http://www.csl.sri.com/projects/nides/

Proctor, P. (2001). *The Practical Intrusion Detection Handbook.* Upper Saddle River, NJ: Prentice Hall

Ringleben, K. (2002, June 21). Intrusion detection systems grow up Retrieved June 19, 2006, From http://searchnetworking.techtarget.com/qna/0,289202,sid7_gci834569,00.html

Schetina, E., Green, K., and Carlson, J. (2002). *Internet Site Security*. Boston, MA: Addison Wesley

Singh, A. (2005, December 14) Demystifying Denial-Of-Service attacks, part one. SecurityFocus. Retrieved November 21, 2006 from http://www.securityfocus.com/infocus/1853

Spafford, E. (n.d.). The *Internet Worm Program: An Analysis* (Purdue Technical Report CSD T8-823). West Lafayette, IN: Purdue University. Department of Computer Science. Retrieved July 5, 2006, from http://homes.cerias.purdue.edu/~spaf/tech-reps/823.pdf

Strebe, M., and Perkins, C. (2000). *Firewalls 24seven*, second edition. San Francisco, CA: SYBEX

Sundaram, A. (1996). An Introduction to Intrusion Detection. Association for Computing Machinery. Retrieved July 4, 2006, from http://www.acm.org/crossroads/xrds2-4/intrus.html Boston, MA: Thomson

Whitman, M. and Mattord, H. (2007). *Incident Response and Disaster Recovery.* Boston, MA: Thomson

Whitman, M. and Mattord, H. (2006). *Readings and Cases in the Management of Information Security.* Boston, MA: Thomson

Whitman, M. and Mattord, H. (2005). *Principles of Information Security*. Boston, MA: Thomson