Global Information Assurance Certification Paper

GIAC Certified Intrusion Analyst Practical

NIDS Change Management & Version Control, Network Detects, and Log Analysis

Bill M. Shinn
Practical Assignment v3.1
September 20, 2002

SANS 2002, Orlando, FL
April 1st- 7th, 2002

Assignment 1 - State of Intrusion Detection 5
Abstract 5
Introduction 5
Challenges 6
Goals and Requirements 8
Recommendations 9
Technical Recommendations 15
Conclusion 24
Assignment 2 - Three Network Detects 27
Detect 1 - Radware's Linkproof & Nmap TCP Scan False Positives 28
Detect Trace Logs 28
Source of Trace 30
Detect was generated by 30
Probability the source address was spoofed 31
Description of the attack 34
Attack mechanism 36
Correlations 38
Evidence of active targeting 38
Severity 39
Defensive Recommendations 39
Multiple Choice Question 40
Detect 2 - VPN Policy Violator - Disabled Split Tunneling Saves the Day 42
Detect Trace Logs 42
Source of Trace 42
Detect was generated by 42
Probability the source address was spoofed 42
Description of the attack 43
Attack mechanism 45
Correlations 45
Evidence of active targeting 46
Severity 46
Defensive Recommendations 47
Multiple Choice Question 47
Detect 3 - DDoS Shaft SYNflood - Weak, Anomaly or Backscatter? 49
Detect Trace Logs 49
Source of Trace 61
Detect was generated by 61
Probability the source address was spoofed 62
Description of the attack 63
Attack mechanism 64
Correlations 64
Evidence of active targeting 65
Severity 65

Defensive Recommendations 66
Multiple Choice Question 66
Assigment 3 - Analyze This! 69
Executive Summary 69
Log Files Analyzed 69
Prioritized Detects 70
Compromised or Malicious Hosts within University Network 70
Attack Participants within University Network 73
False Positives and Irrelevant Alerts or Scan 73
"Top Talkers" 75
Quantitative Alert Analysis 75
Quantitative Scan Analysis 79
Significant Relationship/Link Graphing Technique - 12.151.57.37 & MY.NET.88.245 80
Critical External Hosts 84
Out of Spec Packets 88
Defensive Recommendations 91
Methodology 92

Writing Conventions

To aid the reader in quickly differentiating content, the following conventions are used throughout this practical:

Normal text appears in 12-point TrueType Times New Roman font.

Hostnames and interface names appear in 10-point TrueType Times New Roman Bold font.

Captions to diagrams, in-line hyperlinks, bibliographic citations and other excerpts appear in 9 or 10-point TrueType Times New Roman font.

```
Command text and file names appear in 10-point TrueType Courier
New font.
```

Log files and network traces appear indented in 8-point Times New Roman font.

Assignment 1 - State of Intrusion Detection

A Change Management and Version Control Workflow for Production Snort NIDS Sensors

Abstract

This paper defines a comprehensive change management and version control workflow for maintaining policies on production Snort IDS sensors. Beginning with a theoretical discussion of the challenges and goals for incorporating Network Intrusion Detection Systems in enterprise change and version control processes, an argument is advanced for careful control of NIDS signature and configuration file deployment. Both procedural and technical recommendations are presented to guide organizations through critical phases of sensor policy management, from development to deployment, accounting for roll back and remediation.

Introduction

This paper outlines the basic requirements of a comprehensive change management and version control workflow for policies on production Snort IDS sensors. Consistently, one of the challenges faced in administering Snort sensors is updating and deploying the latest signatures and configuration files throughout the enterprise Network Intrusion Detection System (NIDS), while preserving local control and the granular signature details inherent in a finely-tuned sensor. This paper is intended to help an organization consider the impact of a NIDS on its operational environment prior to implementation, or to assist organizations already confronting recurring issues with NIDS management.

Even the best change control procedures and the most dynamic version control systems may not account for the unique demands of a NIDS, where integrity of data-in-transit, timely testing and deployment, and access controls are of paramount importance. Beginning with a broader theoretical discussion of the challenges and goals in change management and version control applied to network intrusion detection, this paper concludes with a practical workflow that can be employed to overcome obstacles, while still meeting clearly defined objectives.

At the foundation of the challenges involved in maintaining a NIDS is the day-to-day operation that must take place to maintain a working set of configuration files and signatures deployed to each sensor. These files compose a *policy* that must limit false-positive alerts, while detecting the latest exploits applicable to the network being monitored. Building several sensors, installing a default set of signatures, and attaching them to the LAN is relatively simple. Deploying sensors in a managed data center where multiple, geographically dispersed departments and levels of management are responsible for different functional areas of a production environment is a significantly more challenging task. Beyond initial deployment is the constant revision and fine-tuning required for protection of information assets in which the NIDS is designed to assist. This revision and constant change must be conducted according to the procedures standard to an organization and designed to minimize disruption and account for the enterprise's workforce and budget resources.

This paper assumes that any enterprise advanced enough to consider and implement a NIDS has addressed, or is actively addressing, the requirement for a robust change control system for its production/operations environment.[1,2] Such a system can be large or small in scale, from simple forms and log books, to relational databases and wireless notification systems, but it should account for each critical system in the enterprise. The identification of critical systems is relative to each organization, but *minimally* includes network devices such as switches, routers, and firewalls upon which an organization's connectivity and security depend; servers responsible for hosting corporate web sites or enterprise applications; and hosts running data warehousing, ERP or CRM applications. Workstations, management hosts internal to an IT group, and development LANs may or may not be included in the formal "production" controls. This paper also assumes that version control is in place for all code developed within the enterprise and for configuration files required by critical applications and servers. These are rather large assumptions…

While a NIDS offers detection, and sometimes prevention, of attacks, the best security comes from careful systems management and resistance to the unknown. Moreover, without holistic change management, remediation from attack is almost impossible as no accurate baseline of the system configuration prior to compromise exists. Working from the above assumptions, the real challenge then comes from applying and adapting these existing change management and version control procedures to a newer concept such as NIDS.

Although an oversimplified view, many traditional change controlled production environments govern a workflow in which developers or vendors write code and build packages, management approves deployment, systems administrators or mainframe operators deploy or install the package, then an Operations team or support desk monitor the application in production - this cycle continues with the reporting of errors and feature requests from Operations or users to developers or engineers. The contemporary NIDS does not fit well in this model, without adjustment, for a number of reasons.

Challenges

No Clear "Ownership" of Sensors between Security and Operations

The "ownership" of a NIDS sensor, depending on the organization, may not be clearly defined - "ownership" meaning responsibility and accountability for the daily operation and life-cycle of a system. Security Analysts and Security Managers are the owners of the information produced by the sensor, but may not be tasked with the constant monitoring of the sensor (network connectivity, disk space, log rotation, creating accounts, etc), nor the installation and maintenance of the required packages on each sensor (ssh, mysql client, libpcap, etc.). Prior to the rollout of each NIDS sensor, the ownership of maintenance and monitoring tasks must be clearly defined and incorporated into daily schedules, budgets, and change control procedures.

To illustrate, which department is responsible for proposing configuration changes to a NIDS sensor? Which team or individual is tasked with testing the change on a lab sensor prior to turnover into production? Which department actually implements the change? The answer to this question may depend on the type of change, and certainly upon the structure of an organization. For an upgrade or patch to stunnel or OpenSSH, the organizational knowledge may reside in the Operations department. For an upgrade to the Snort program or the signatures, the Security Analysts may want to compile the code themselves and edit .rules files manually. Who owns each task is not as important as ensuring that *someone* owns the task and is equipped with the training and resources to manage it.[3]

No Clear Separation of Roles between "Development" and Operations
The separation of roles and responsibilities between developers and Operations staff - the core of many change control models - is blurred when applied to NIDS. Additionally complicated are the access rules applied to a sensor, as "developers" are (or should be) typically restricted from accessing production hosts. Inherent in NIDS deployments, is the "niche knowledge" required to maintain a working policy. If the Security Analyst is seen as a "developer" of signatures and sensor policy, there should logically be a member of the operations staff that deploys the "code" (signatures and .conf files). In most Snort deployments, however, the Security Analyst probably logs directly into the sensor and either uploads, or directly edits the Snort signatures because he or she is the only person who knows how. Depending on the Snort setup, this person may also need root access to restart the daemon. This is a classic problem, not just with IDS, but with all applications requiring the direct support of a developer with a highly specified skill set. While many organizations may "hand-off" a product to an Operations team equipped with programmers capable of troubleshooting run-time errors, an Operations teams may not be equipped with a dedicated Security Analyst empowered and trained to fix a rule producing 1000 false-positives every 5 seconds. [4]

This is compounded by the lack of a versioning system for Snort policies which accommodates the many changes an organization *should* make in fine-tuning and maintaining signatures and configuration files.[5] Granted, that both the current and stable distributions of the Snort signatures can be tracked using CVS, but this alone does not provide a mechanism for an organization to test, modify, package, and deploy the updated signatures to production sensors.

Additional Security & Audit Requirements of NIDS Sensors
Placing additional stress on change management systems is the heightened burden of proof imposed upon IDS data used in forensic, and the commensurate integrity required of such systems producing this data - forcing organizations to severely limit access to IDS sensors. This is at odds with the needs of many IT departments to cross-train their employees as a hedge against employee turnover or downsizing, and to provide adequate coverage during second or third shifts. Organizations must now introduce levels of access control, accountability, and procedural controls acceptable to the scrutiny of IT audits which may increasingly include examination of NIDS implementation and management.

Change management practices and version control processes which include NIDS must also provide proper evidence that rule set integrity is not compromised, that access to sensors and their data is logically controlled by job function, and that changes to a production IDS environment are authorized and controlled.

Goals and Requirements

In response to these challenges, any proposal of a comprehensive workflow for change management and version control of Snort NIDS sensors must meet a lengthy list of goals and requirements:

- Always have available, and ready for review, the current version of the stable signature and configuration files from the official Snort distribution. Although every change to the official Snort stable rule set is not applicable to every environment, every Snort deployment should have signatures at its disposal to detect the latest exploits and to incorporate enhancements to signatures. This should be considered "pristine" source code and never be placed into production without testing and alteration to an operational environment. For instance, signatures applying to Microsoft SQL should not be deployed to protect Oracle-only environments, as this can only burden Snort without the benefit of intrusion detection.[6] Also, as each sensor should be running a policy fine-tuned and trimmed to the monitored network, each policy will typically have different variables in snort.conf.

- Any change management and version control system should provide notification when the official stable rule set distribution changes, or when an analyst or member of security team submits a change proposal from testing to be deployed onto the production sensors.

- A fundamental feature of change management procedures is preserving the ability to review and approve changes to all sensor policies *before* introducing into production or Operations environment. This approval should come from at least two parties: the department or person(s) tasked with testing and certifying that a change will not disrupt the production sensor; but also from the party responsible for accepting and turning over the change in production. Accountability, acceptance, and "buy-in" are critical. How this approval takes place - using managers' signatures or log timestamps, for instance - can be taken from existing change management systems, but should occur. [7]

- Separate, but equal, from the approval process - a NIDS change management workflow must include change control documentation which outlines the change that will be made, in addition to the approval itself. This must accompany the approval process so that management or those accountable are aware of the actual change. A reference to a log book entry detailing the proposed change, to a tracking system ticket, or to documentation of the actual change (file printout) are all sufficient as long

as the review takes place.[8]

This documentation affords the opportunity for review, but also journals the changes to a sensor so that any required remediation can take place. This step mitigates risk of improper sensor operation caused by deployment of unintended signature or configuration file changes, allowing the suspect change to be isolated and rolled back.

- Primarily the role of a version control system, a comprehensive workflow for NIDS management must mitigate risk of improper sensor operation caused by multiple analysts editing common rule sets in a distributed environment. This version control system, distinct from, but integral to the change control process, must allow for concurrent work on the same policies by multiple analysts. This system enforces the need to approve changes before turnover, as all parties must commit their changes prior to release.

- Change and version management are fundamentally designed to prevent disruptions and to allow timely remediation from improper sensor configuration. This objective requires more than just the previously mentioned documentation: a build and deployment process allows roll back to previous versions of deployed "code"… in the case of a sensor policy, this "code" is the signatures and configuration files.

- A robust workflow should enforce separation of roles between Security Analysts or Engineers and the Systems Administrators or Operations staff. This philosophy assumes a great deal about the organizational structure and workforce resources of an IT department, and that distinct job titles exist to fulfill each roles. In many environments, the Systems Administrator is the Security Analyst and perhaps the one-person Operations team as well. In small- to medium-sized enterprises, the distinction is no less important as the "small shop" can better account for time-on-task and resource hours when these roles are kept separate…at least conceptually.[9]

- The workflow should be standardized throughout the enterprise to allow quick and seamless addition of new sensors. Any change management and version control systems needs to scale properly for a large number of sensors.

Any system must result in the smooth transition between development, testing and production environments. The fast-paced nature of changes to production Snort NIDS sensors can tax any change control system that fails to account for the unique needs of NIDS operational requirements, even systems which account for emergency changes. Increasing the number of change control windows for NIDS sensors may alleviate some of the burden on the system, as long as the approval, acceptance, and turnover procedures are clearly defined.

A proactive workflow that is widely recognized and accepted by all parties involved in NIDS management can elevate the perceived importance of NIDS to that of other

traditional production systems, deserving of in-depth monitoring, change control procedures, and resources.

Recommendations

Combining manual tasks and automation services, the workflow recommended in this paper attempts to leverage the most productive, affordable tools available to manage Snort sensors, regardless of platform availability. While potentially daunting at first, the tools defined are relatively trivial to set up (or may already be in use), and once established allow the workflow to scale properly to a large number of sensors. The role of each tool is described later in the more technical section of this recommendation. Sections of the recommendation can benefit from overall refinement and most likely require adaptation to a particular environment. Although this recommendation attempts to meet each of the goals described above, every environment is different due to resource limitations, technical capacity, or commitment.

The tools used to manage an official distribution of the Snort rule set, modify it for a given environment, obtain approval and ensure documentation, and deploy (or roll back) a build are illustrated in Figure 1.

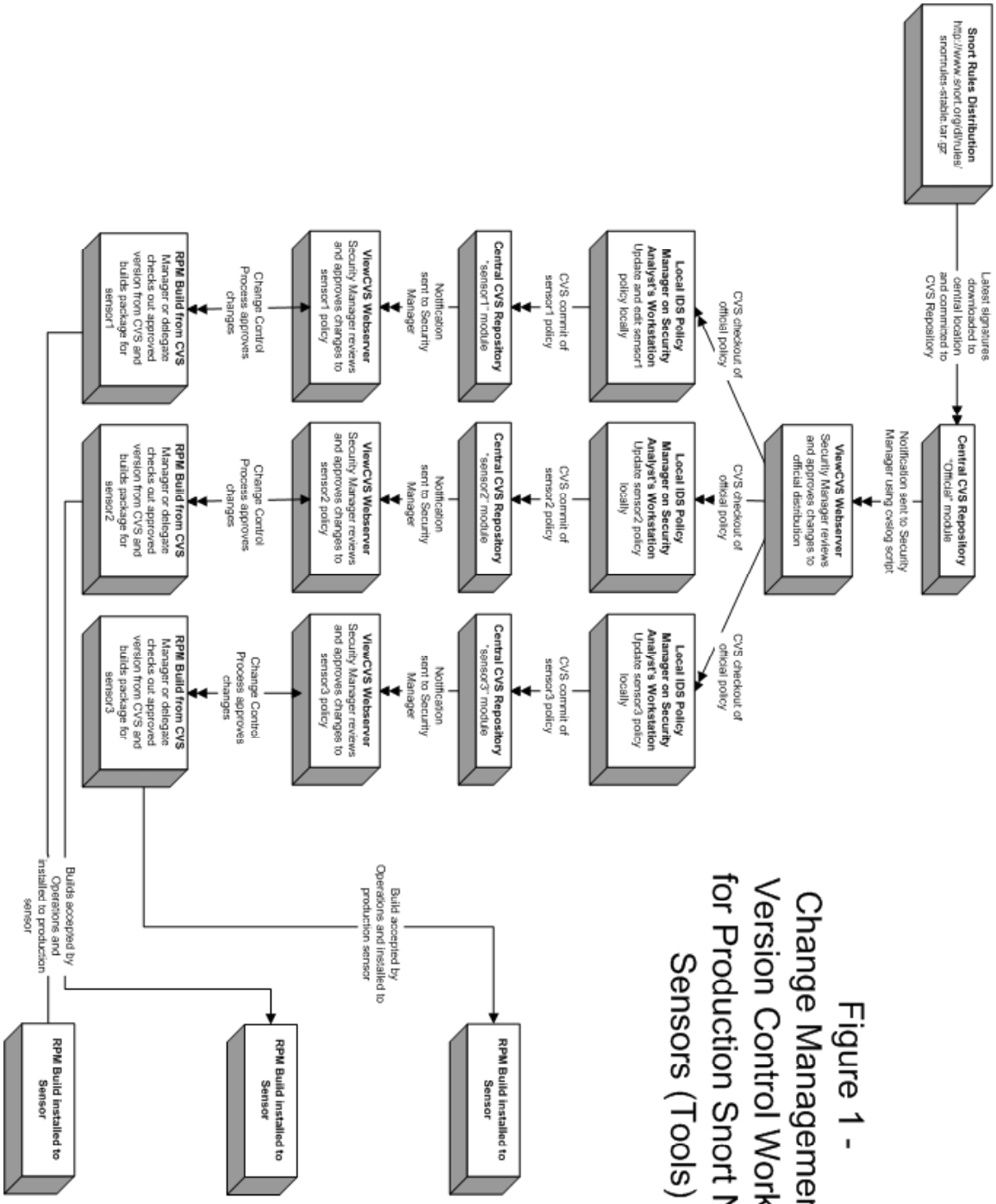


Figure 1 - Change Management & Version Control Workflow for Production Snort NIDS Sensors (Tools)

Figure 1

Below, I define in greater detail the several positions or individuals within an organization responsible for particular portions of the recommended NIDS change management and version control workflow. As mentioned previously, each play a key role, and intentionally hold separate responsibilities to enhance access control and divide labor. In some cases, one person may occupy all of these roles; the purpose is to identify the separation of roles required for a meaningful workflow. Again, in small to medium organization, one person fulfilling each role can still divide their time and functions evenly and preserve the integrity of the process.

Security Manager
This position or individual is responsible for identifying and justifying each sensor, working with the network engineering staff to determine the best placement on the network, and acting as a liaison between the Security Analysts and the Operations team or administrators. In the day-to-day tasks described in this workflow, the manager is responsible for setting up the IDS Policy Manager to account for all sensors. The manager uploads this information to a central repository and keeps the information up to date, adding and removing sensors when required. The Security Manager also ensures the availability of the other tools required for the success of the workflow.

The Security Manager holds the additional role of downloading the Official Snort signature distribution. The Security Manager receives notification whenever an Analyst proposes a change to the signatures or configuration files on production sensors. If the official signatures change, the Manager notifies the Analysts of the change so the updates can be reviewed and incorporated into testing and development. If an Analyst commits a proposed change to the central repository, the Manager reviews the change and initiates the change control procedures required to update the build on the production sensor. Once released, documented, and approved from the change control process, the Manager (or a delegate) builds the release package that is distributed to Operations for turnover to the production sensors.

Security Analyst
In this workflow, the analyst is not only tasked with interpreting events of interest produced by the NIDS, but is responsible for adapting "vendor" signatures to the proprietary environment of an organization. This position builds custom local signatures, refines the official Snort signatures and configuration files for optimum performance and applicability, and submits these efforts for rollout on the production network. This position works closely with the Security Manager to define the security needs of each business environment. Through either direct contact, or using the Security Manager as a liaison, the Analyst closely tracks changes to the production network (servers, switches, firewalls, ports, etc.) that may impact the operation or performance of the NIDS. If the Analyst is also responsible for regular penetration and vulnerability assessment, the data produced from these events can be used to track changes to the network and adjust NIDS policies accordingly.[10]

Operations or Administration Staff

These positions are responsible for the administration of the sensor machines themselves. They update software and keep the machines running according to best practices (latest security patches to software, most recent libraries, adding and managing accounts, etc.), in addition to monitoring the availability and performance of the sensors. After release from change control, the Operations team is responsible for installing the latest build/package which updates the Snort policy. The Snort binaries, beyond the scope of this paper, are tested by the Analysts in a lab environment, and then turned over to production in a similar manner as the policies.

Finally, prior to the technical portion of this paper, each step of the policy change control process is depicted in Figure 2. The workflow process generally begins with the Development phase for new Snort deployments, but if your enterprise is adopting a change management workflow, or applying one to NIDS for the first time, the "beginning" of the cycle is relative to the progress or state of your deployment.

The first question one might ask is why, in a well-designed change control and version management workflow, are roll back and remediation identified as critical processes? It is assumed that in the life of every NIDS deployment, improper configurations and/or incorrectly authored signatures will be deployed more than once. Despite every best effort in development and testing, signatures deployed to monitor a production network may not produce the desired result, returning too many false positives or resulting in false negatives when confronted with a comprehensive test against the NIDS. At all cost, resist the overwhelming urge to "fix-on-the-fly" and simply bypass all the controls, log into the production sensor, and simply "tweak" the signature or policy setting. Proper roll back according to procedure is perhaps the most difficult, yet most important, step in change and version control for NIDS sensors. To preserve any separation of responsibilities between Analysts (a.k.a - "developers") and the Operations staff, and to preserve the integrity of access controls, proper roll back and remediation involve removing the most recent build of the production policy and signature set, and restoring the last known-good installation - continuing the cycle. The Security Manager works with management peers on the Operations side to enforce these practices.

Figure 2 - Change Management & Version Control Workflow for Production Snort NIDS Sensors (Concepts)

Figure 2

Technical Recommendations

As shown in Figure 1, the technical requirements of the workflow involve a number of tools installed on several distinct sets of hosts:

- *Management Server* - A central repository running a web server, CVS with the cvslog notification script, package management tools such as RPM, and an SSH daemon.
- *Security Manager Workstation* - A Windows 2000 or XP workstation with CVSNT, OpenSSH for Windows, regdmp.exe and regini.exe from the Windows 2000 Resource Kit and IDS Policy Manager. Version 5.6 of the Windows Script Host is required.[11]
- *Security Analyst Workstation* - A Windows 2000 or XP workstation with CVSNT, OpenSSH for Windows, regdmp.exe and regini.exe from the Windows 2000 Resource Kit and IDS Policy Manager. Version 5.6 of the Windows Script Host is required.
- *Snort Sensors* - The Snort sensors can be running on almost any platform compatible with the build/packaging system you choose.

Endnotes referencing required and optional tools are inserted as each tool is discussed; the home page or download location for each application, and referral to useful documentation, is provided. Most projects have very apparent links to download locations and additional documentation resources. Every effort has been made to include configuration information specific to this workflow.

Management Server Setup

This server can be any platform capable of hosting the applications and accounts as described below. This device can be the same server hosting other centralized Snort management applications such as ACID or SnortSnarf, but can certainly be hosted separately. Depending on the separation of resources within any given IT department, the tools below may already exist somewhere in your infrastructure and only require minor changes to accommodate these requirements.

Install and configure SSH Daemon and Accounts

The Security Manager and each of the Security Analysts will need a login account on the management server hosting the CVS modules described below. As both CVS operations and any required interactive logins take place using SSH, each account could have an RSA key generated, and the public key placed in the `/home/username/.ssh/authorized_keys` file for each user. This adds an additional layer of security and convenience to the procedures. Private keys, if they don't already exist, can be distributed off-line.[12,13,14]

Install and configure CVS Repository

Many resources on this subject exist, and if your network already has a CVS server you may request or provision access and space for use by this workflow.[15,16,17,18] This CVS server can reside on nearly any platform, but each Security Analyst and the Security Manager will need account access on this server, with the ability to add CVS modules delegated to at least the Security Manager.

To take advantage of the change notification features described in this workflow, which greatly enhances the management of sensor policies, the CVS server should be of the *nix variety with Perl, diffstat, and sendmail installed. The cvslog script, which provides this notification feature, requires these applications for functionality.[19] Prior to creating the CVS module for each sensor and the official distribution, install and configure cvslog according its instructions. Again, this is not required, but without it notification of changes to the policies will not take place. Ideally, configure the `loginfo` file required by cvslog to send full diff's during each notification, as this provides very granular detail of changes.

Install and configure ViewCVS and an HTTPS server

Installation of both a web server and ViewCVS are well beyond the scope of this paper. [20] Both provide an important piece of the management workflow outlined in this paper, and while not vital, their absence will make the change control reporting and remediation processes much more difficult. ViewCVS offers an easy-to-use interface for generating change control documentation for presentation to a change control committee or for use in the approval process.

The SSL-secured web server hosting ViewCVS will display the CVS modules used in this management workflow. The only setup of ViewCVS proprietary to this recommendation is ensuring that ViewCVS points to the correct `CVSROOT` path which includes the modules for each sensor, the module for the official Snort rules distribution, and the module for the IDS Policy Manager registry settings that need to be installed on all participating workstations. Access to this web or virtual directory on the server should be limited to Security Analysts and the Security Manager.

Package Building Tools

You will need package building software compatible with the build platform and the sensors. In the testing environment for this workflow, both the management server platform and the sensors ran on Red Hat 7.x hosts, so RPM was used exclusively.[21] With adjustment and the right skill sets, the packaging could occur on Debian and Solaris platforms as well, or even using a solution such as OpenPKG.[22,23] At a minimum, the Security Manager will need rights on this server to create packages; should the Manager

delegate this responsibility, the Security Analysts will also require access to these tools.

Once the signatures and configuration files developed and refined by Analysts are released from the change control process, the approved release of each policy is checked out from CVS and a package is built for installation to the sensor in question.[24,25,26,27] The process of creating packages takes place on this server or any other workstation with sensor-compatible build tools. As building a policy package does not require anything to be compiled; only a standard set of file manipulation tools is required. If your organization runs Snort on Win32 platforms, additional research needs to be conducted on building packages for these systems.

While almost all of the CVS commands required for managing the sensor policies using IDS Policy Manager on workstations has been scripted into `ipmadmin.vbs`,[28] some knowledge of CVS is useful for building the packages. Building and using policy packages is described in greater detail following the discussion of setting up the workstations.

Security Manager Workstation Setup

Install and Configure CVSNT & Network Simplicity OpenSSH for Windows

Follow the standard installation instructions for both applications.[29,30] For CVSNT, choose the custom installation option. Select to install only the command line client, password protocol, and external command protocol. For OpenSSH, install client utilities only and be sure to run `ssh-keygen` from the `c:\Program Files\NetworkSimplicity\ssh` directory, which will generate the required `.ssh` subdirectory. Please any private RSA keys required to log into the central server in `C:\Program Files\NetworkSimplicity\ssh\.ssh`. Also be sure to connect from the command line to the central server at least once so the fingerprint is added to the `known_hosts` file successfully and to ensure the connection works properly.

Ensure that the following are permanently added to your path:

```
c:\"Program Files"\"CVS for NT"
c:\"Program Files"\NetworkSimplicity\ssh
```

Installing ipmadmin.vbs

From the command shell or Windows Explorer, change to the root of the drive (or other optional location) and create a directory titled `ipmadmin` for the `ipmadmin.vbs` script to run from. Copy the script to this location:

```
>mkdir c:\ipmadmin
>copy [x:\ | \\]path\to\ipmadmin.vbs c:\ipmadmin
```

Open the file in a text editor and modify the required and optional user variables

necessary to customize the script to your environment.

Install and Configure IDS Policy Manager

Follow the standard installation procedures.[31] Do not configure any sensors or policies at this time. In the `C:\ipmadmin` directory, create one subdirectory entitled `official` and another subdirectory for each of the sensors you wish to manage (naming each policy with the sensor's host name is a helpful convention. Use only the host name, not the FQDN as the "." cannot be used to name the CVS modules based on the sensor policy names). For instance:

```
C:\ipmadmin\official
C:\ipmadmin\sensor1
C:\ipmadmin\sensor2
```

During the initial setup, you will need to populate each policy folder with signatures and configuration files before entering additional details in IDS Policy Manager (IDS Policy Manager requires that `snort.conf` be present before it will create the policy). If you are deploying sensors for the first time, you can untar the `snortrules-stable.tar.gz` file from Snort.org into each directory (moving the files from the default "rules" subdirectory to the root of the policy's folder), or if you are bringing existing sensors into this workflow, you can copy the current production signatures and configuration files into the appropriate directory for that sensor policy (each time another sensor is taken under management, a new directory will need to be created to hold the policy). In this workflow, keeping all signature and configuration files in one directory simplifies management - just ensure that the `RULES_PATH` variable in `snort.conf` reflects that all signatures are in the same directory. Similarly, because in this workflow `snort.conf` is maintained using CVS with all other files, it's easier to install this file to the same location on the sensors as the signatures, being careful to start the Snort daemon with `-c` switch pointed at the right path to `snort.conf` (e.g. - `/usr/local/snort`). This differs from many Snort installations where `snort.conf` is in the `/etc` or `/usr/local/etc` directories.

Using IDS Policy Manager according to the instructions, create the policy for each sensor in your network. By default, IDS Policy Manager creates its own "Official" policy - delete this and create your own with using the path above. This workflow does not use the "Sensors" tab of IDS Policy Manager as this is used to upload directly to the sensors; in this workflow, policies are reviewed, release packages are built, then the packages are turned over into production by Operations staff.

The *Add New Policy* dialog in IDS Policy Manager allows you to choose the *Policy Name*, again use the DNS host name or a standard naming convention consistent with the folder names - it is very important that the folder names and the names of the policies match, as this will coincide with the names of the CVS modules for each sensor policy. Specify the location(s) of the policy as:

```
C:\ipmadmin\official\snort.conf
C:\ipmadmin\sensor1\snort.conf
C:\ipmadmin\sensor2\snort.conf
```

Enter a description of the sensor that will help identify its placement or role. For your actual sensor policies, change the *Update Policy From > Policy Location* selection to "Local." This will expose the *Policy Name* drop-down list; choose "official". Leave all other settings to their defaults.

Initialize the CVS module for IDS Policy Manager Settings

From the command shell

```
>cd c:\ipmadmin
>cscript ipmadmin.vbs /c
```

When this is run, the Visual Basic script retrieves all IDS Policy Manager values from the registry, using tools from the Windows 2000 Resource Kit, and stores them in the following file:

```
c:\ipmadmin\ipmsettings\ipmsettings.reg.
```

The command also runs, automatically, the following CVS command that creates a CVS module to host the registry settings for IDS Policy Manager. These settings are redistributed throughout the network to each Security Analyst:

```
>cvs -d :ext:username@cvsserver.yourcorp.com:/path/to/cvsroot /
import /ipmadmin ipadmin ipadmin_0-1
```

Finally, create a CVS module for the official distribution and each sensor policy:

```
>cscript ipmadmin.vbs /s official /p /r n_n_n
>cscript ipmadmin.vbs /s sensor1 /p /r n_n_n
>cscript ipmadmin.vbs /s {sensor policy name} /p /r n_n_n
```

This `/r` switch followed by a tag is required when importing files into a CVS module for the first time. The naming convention is based on the major Snort release number, followed by the minor release, followed by the release of the signatures in your enterprise. According to this convention, the official signatures and configuration files are “release zero”. The script adds additional variables as it runs, so the resulting tag in CVS is {sensor policy name}-policy_{snort major version}_{snort minor version(s)}_{policy release number}. By example, a sensor policy named “watchdog” using a policy for Snort version 1.9, the tag in CVS would be *watchdog_policy-1_9_0* for the initial import. If the sensor were still running Snort version 1.8.6, the value passed to the `/r` switch could be `1_8_6_0`, resulting in a CVS tag of *watchdog_policy-1_8_6_0*. This naming convention allows for an easy transition to the build process where RPM packages will be named for

the release tag in CVS for each policy.

When sensors are added to the network, the Security Manager configures them locally in IDS Policy Manager as before, and then uses the following command to reflect the changes in the central repository:

```
>cscript ipmadmin.vbs /e
```

Security Analyst Client Setup

Install CVS for NT and Network Simplicity OpenSSH on Windows using the instructions for the Security Manager, taking care to add the applications to your path, generate keys and test connections in the same fashion. Also complete the instructions for copying the `ipmadmin.vbs` Visual Basic script to the workstation, using the same path used to set up the Manager's workstation and setting any user-specific variables in the script.

Installing and Configuring IDS Policy Manager

Because the Security Manager has already configured the IDS Policy Manager settings, and uploaded the settings to CVS - along with the official and any existing policies - the Analyst need only check out the latest settings using the CVS functionality built into `ipmadmin.vbs`.

Install IDS Policy Manager using the standard install process. Following the installation, run the following from the command line (the final command switch is a lower-case "L" as in "local"):

```
>cd c:\ipmadmin
>cscript ipmadmin.vbs /i
>cscript ipmadmin.vbs /l
```

This command sequence checks out the "ipmsettings" module from the central CVS repository and imports the registry settings for the policies configured by the manager - this eliminates the need for an Analyst to set up each policy locally and ensures that all policies are configured the same throughout the organization. The final command displays the sensor policies configured by the Manager that were downloaded from the CVS repository. The Security Analyst reviews the list of policies, then checks out those in need of refinement. Prior to checkout, the signature and configuration files in the policy can be reviewed in ViewCVS.

Editing Policies and Submitting Changes

Use the following commands to check out a specified sensor policy into a working directory:

```
>cd c:\ipmadmin
>cscript ipmadmin.vbs /s {sensorname} /cco
```

After checking out one or more policies from CVS, those policies are available to use in IDS Policy Manager. Using the rich graphical interface, make changes to the policy as required. Import official signatures from the working directory created by checking out the "official" distribution module from CVS. Add `.rules` files using the tools available within IDS Policy Manager - this ensures that the include statements within `snort.conf` are updated; however, when adding a file, also issue the following command to specify that the file should be uploaded to the CVS repository during the next commit:

```
>cd c:\ipmadmin
>cscript ipmadmin.vbs /s {sensorname} /ca {filename.rules}
```

To remove a file, first use the tools within IDS Policy Manager, but also delete the file from the local file system, then issue the following command to schedule the file to be removed from CVS during the next commit.

```
>cd c:\ipmadmin
>cscript ipmadmin.vbs /s {sensorname} /cr {filename.rules}
```

When complete, issue the following from the command-line:

```
>cd c:\ipmadmin
>cscript ipmadmin.vbs /s {sensor policy name} /cci
```

This will commit the changes to CVS - first prompting for comment in a text editor - and notify the Security Manager (and those set up to receive notifications in cvslog) of the change. When satisfied that a checked out set of files constitutes a viable release for production, issue the following to tag the files with a release number:

```
>cd c:\ipmadmin
>cscript ipmadmin.vbs /s {sensorname} /t /r n_n_n
```

In this case, the value of the `/r` switch is the release number using the same naming conventions discussed earlier. Using ViewCVS, a Security Manager or other interested parties, can easily create documentation for change control review based on the differences between files tagged with the release number and those currently in production (based on the last deployed release number in the repository).

Updating the "official" Module

Because each *File > Save* in IDS Policy Manager modifies the date on all files checked out to a working directory, this can create some confusion as to which files actually changed during a CVS commit. Although this is easily resolved by reviewing diff's of

each version, it is best to avoid this confusion with the official Snort distribution. Therefore, using CVS commands directly preserves the idea of "pristine sources" that vital to CVS. To update the stable rules, download the most current stable distribution, unpack it (this can be done from any platform, but Linux usually has these utilities available). In this example, the update occurs from the Security Manager's home directory on the central server):

```
$wget http://www.snort.org/dl/rules/snortrules-stable.tar.gz
$tar xzvf snortrules-stable.tar.gz
$cvs -d /path/to/cvsroot checkout official
$ls official > official.list
$ls rules > rules.list
$diff rules.list official.list > snort.diff
$mv rules/* official
$rmdir rules
```

If files were removed or added from the official distribution, you need to add or remove them from the CVS working directory explicitly. To perform this, you will need to know which files were in the "official" working directory prior to overwriting them with the new stable release and must also have a way to determine which files are no longer included in the official distribution. That is why, following each update from Snort.org, the difference file is created between the newly unpacked files and those currently in the repository. By reviewing the contents of `snort.diff`, you will know which files to manually remove from CVS (first deleting the file, then issuing the `cvs remove` command) and which to add (using the `cvs add` command). This review could probably be scripted, but since the addition or removal of files from the stable distribution is fairly infrequent, there is only an occasional need to pass the required CVS commands manually. Once the working directory for the official distribution is up to date, commit it to the repository:

```
$cvs -d /path/to/cvsroot commit official
```

Build Process

In order to build a package to deploy each sensor policy, a source "distribution" and an RPM .spec file need to be created from which to build the final package. After concluding the change control approval process, the release number is used to checkout the approved instance of the policy and set up the naming conventions for the RPM package. In the example, a package is being created for a sensor named "watchdog" running Snort version 1.9. Release "1" of the sensor policy is being deployed; operations take place from the Security Manager's home directory (~secmanager):

```
$cvs checkout -r watchdog_policy-1_9_1 watchdog
$tar -cf watchdog_policy-1_9_1.tar watchdog
$tar -f watchdog_policy-1_9_1.tar --delete watchdog/CVS*
$cvs release -d watchdog
```

Depending on the access rights configured on the management server, the following may need to be performed as root:

```
# mv ~secmanager/watchdog_policy-1_9_1.tar /
/usr/src/redhat/SOURCES/
# cd /usr/src/redhat/SPECS
# vi watchdog_policy.spec
```

Add the following to the .spec file, noting that the install path may vary depending on the sensor configuration and platform, as may the default variables used by RPM. Also note that the "release number" used to tag the sensor policy is the last number in the "Version" entry below; the "Release:" number in an RPM .spec file is specific to the release of this RPM package, not the software being packaged:

```
Summary: Snort Policy
Name: watchdog_policy
Version: 1_9_1
Release: 1
Copyright: GPL
Group: Applications/IDS
Source0: %{name}-%{version}.tar
Packager: Security Manager
BuildRoot: %{_builddir}/%{name}-root

%description
Snort signatures and configuration files for corporate sensor
watchdog

%prep
rm -rf $RPM_BUILD_ROOT/watchdog

%setup -n watchdog
%build

%install
mkdir -p $RPM_BUILD_ROOT/usr/local/snort
install $RPM_BUILD_DIR/watchdog/*
$RPM_BUILD_ROOT/usr/local/snort
%clean
rm -rf $RPM_BUILD_ROOT
rm -rf $RPM_BUILD_DIR/sentry3
%files
%defattr(600,snort,snort)
/usr/local/snort
```

Once the .spec file is complete, build the package and review the contents (copying the `.spec` is optional):

```
# rpm -bb -vv watchdog_policy.spec
# rpm -qlp ../RPMS/i386/watchdog_policy-1_9_1.i386.rpm
# cp watchdog_policy.spec ~secmanager
# exit
```

Optionally - Store the .spec file in CVS for later modification.

```
$ cvs -d /path/to/cvsroot checkout watchdog
```

```
$ mv watchdog_policy.spec watchdog
$ cd watchdog
$ cvs add watchdog_policy.spec
$ cd ..
$ cvs -d /path/to/cvsroot commit watchdog
```

Copy the RPM to a standard location on the sensor. The command below is run from the build server by a member of the Operations team, this eliminates the need for the Security Analysts or Security Manager to have an account on each sensor (enforcing separation of duties and divided access controls).

```
$ scp /usr/src/redhat/RPMS/i386/watchdog_policy-1_9_1 /
operations@watchdog.yourcorp.org:/home/operations/RPMS
```

Finally, a member of the Operations team installs or upgrades the package on the sensor, using normal RPM commands, and restarts the Snort process. Verification that the change took place can be reported with file and system integrity tools, in addition to the built-in RPM query functions. If problems occur and remediation is required, simply use the "upgrade" features built in to RPM to roll back to the last known good configuration. For example, if you upgraded from watchdog_policy-1_9_1-1 to watchdog_policy-1_9_2-1 and wanted to roll back, issue the following command on the sensor:

```
#rpm -Uvh --oldpackage watchdog_policy-1_9_1-1.i386.rpm
```

When a new release of the sensor policy is required and approved, repeat the packaging process, substituting the correct Snort version and policy release number.

Conclusion

While the tools used in the Technical Recommendation section assist the process of change management and version control, by no means are they exhaustive or required for a meaningful workflow. In some environments, or for some Managers and Analysts, other tools or methods are preferred; the importance is the process itself. Until IDS vendors incorporate change management and version control, which maintains and accounts for the large degree of customization inherent in NIDS deployments, tools and practices such as those in this paper are required to create a robust system for reducing the risk of improperly configured sensor policies.

Most organizations practice at least minimal change management of critical security devices such as firewalls; and often include the security offered by such devices when marketing their products and services. If the security of your organization is part of its overall business and marketing strategy, and your partners and clients rely on claims of a well-implemented NIDS when making business commitments, then a NIDS deserves at least the same level of controls as other systems. Without the configuration integrity, scalability, and remediation capacity offered by the added layer of a comprehensive

change management and version control workflow, the potential effectiveness of Network Intrusion Detection Systems is limited.

Assignment 2 - Three Network Detects

Network Diagram - This diagram depicts the network used for all three detects.

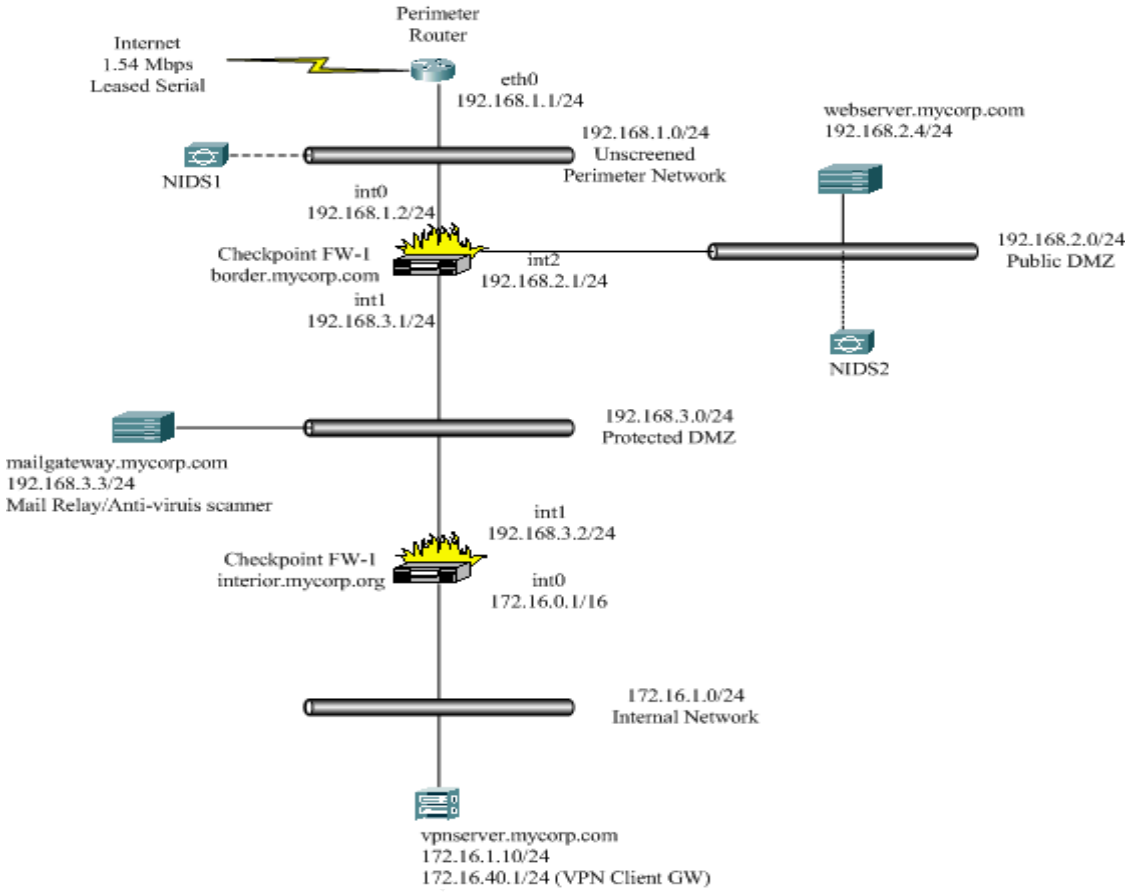


Figure 1: Network Diagram

Summary of hosts and devices:

edge.mycorp.com - Cisco 2500 series router using standard anti-spoofing ACLs
nids1.mycorp.com - Snort 1.86 IDS sensor on RedHat Linux 7.2. Promiscuous interface is not assigned an IP address.
border.mycorp.com - CheckPoint Firewall-1 with three interfaces. int0 faces the public Internet, int1 faces a protected DMZ hosting only the mail relay and the connection to the Corporate LAN, and int2 provides the default gateway for the public DMZ hosting a corporate web server.
webserver.mycorp.com - Windows NT Server 4.0, SP6a running IIS 4.0 hosting a public web site.
nids2.mycorp.com - Snort 1.86 IDS sensor on RedHat Linux 7.2. Promiscuous interface is not assigned an IP address.
mailgateway.mycorp.com - Windows 2000 Server, SP3 running an SMTP Virtual Server and GFI MailEssentials for anti-virus and other email filtering. This relays mail to the internal network.
interior.mycorp.com - Checkpoint Firewall-1 with two interfaces. int1 faces the protected DMZ and int0 provides the egress point for the internal corporate network.
vpnserver.mycorp.com - Windows 2000 Server, SP3 providing RRAS services to VPN clients. Connections are tunneled through the firewalls and terminated at this server. VPN clients are assigned an address on the 172.16.40.0/24 subnet and use the VPN server's secondary IP address as a default gateway for the tunneled connection.

Detect 1 - Radware's Linkproof & Nmap TCP Scan False Positives

Detect Trace Logs

Snort alerts from **nids1.mycorp.com**

Jun 6 12:37:32 nids1.mycorp.com snort[31510]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.130.21:80 -> 192.168.2.4:80
Jun 6 12:37:32 nids1.mycorp.com snort[31510]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.135.21:80 -> 192.168.2.4:80

Jun 13 08:10:10 nids1.mycorp.com snort[2025]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.130.21:80 -> 192.168.2.4:80
Jun 13 08:10:10 nids1.mycorp.com snort[2025]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.135.21:80 -> 192.168.2.4:80

Jun 25 15:24:26 nids1.mycorp.com snort[783]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.130.21:80 -> 192.168.3.3:**25**
Jun 25 15:24:26 nids1.mycorp.com snort[783]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.135.21:80 -> 192.168.3.3:**25**

Jun 28 08:39:58 nids1.mycorp.com snort[10281]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.130.21:80 -> 192.168.2.4:80
Jun 28 08:39:58 nids1.mycorp.com snort[10281]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.135.21:80 -> 192.168.2.4:80

Snort logs from **nids1.mycorp.com**

```
[**] SCAN nmap TCP [**]
06/06-12:37:32.176220 199.197.130.21:80 -> 192.168.2.4:80
TCP TTL:53 TOS:0x0 ID:3897 IpLen:20 DgmLen:40
***A**** Seq: 0x3FD  Ack: 0x0  Win: 0x578  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] SCAN nmap TCP [**]
06/06-12:37:32.206220 199.197.135.21:80 -> 192.168.2.4:80
TCP TTL:52 TOS:0x0 ID:3900 IpLen:20 DgmLen:40
***A**** Seq: 0x3FF  Ack: 0x0  Win: 0x578  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] SCAN nmap TCP [**]
06/13-08:10:10.271059 199.197.130.21:80 -> 192.168.2.4:80
TCP TTL:53 TOS:0x0 ID:45533 IpLen:20 DgmLen:40
***A**** Seq: 0x378  Ack: 0x0  Win: 0x578  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] SCAN nmap TCP [**]
06/13-08:10:10.311059 199.197.135.21:80 -> 192.168.2.4:80
TCP TTL:52 TOS:0x0 ID:45536 IpLen:20 DgmLen:40
***A**** Seq: 0x37A  Ack: 0x0  Win: 0x578  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] SCAN nmap TCP [**]
06/25-15:24:26.890538 199.197.130.21:80 -> 192.168.3.3:25
TCP TTL:53 TOS:0x0 ID:41685 IpLen:20 DgmLen:40
***A**** Seq: 0x3D3  Ack: 0x0  Win: 0x578  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

[**] SCAN nmap TCP [**]
06/25-15:24:26.920538 199.197.135.21:80 -> 192.16.3.3:**25**
TCP TTL:54 TOS:0x0 ID:41688 IpLen:20 DgmLen:40
A* Seq: 0x3D5 Ack: 0x0 Win: 0x578 TcpLen: 20

=+

[**] SCAN nmap TCP [**]
06/28-08:39:58.912724 199.197.130.21:80 -> 192.168.2.4:80
TCP TTL:53 TOS:0x0 ID:16918 IpLen:20 DgmLen:40
A* Seq: 0x363 Ack: 0x0 Win: 0x578 TcpLen: 20

[**] SCAN nmap TCP [**]
06/28-08:39:58.952724 199.197.135.21:80 -> 192.168.2.4:80
TCP TTL:54 TOS:0x0 ID:16921 IpLen:20 DgmLen:40
A* Seq: 0x365 Ack: 0x0 Win: 0x578 TcpLen: 20

=+

Snort alerts from **nids2.mycorp.com**

Jun 6 12:38:20 nids2.mycorp.com snort[24288]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.130.21:80 -> 192.168.2.4:80
Jun 6 12:38:20 nids2.mycorp.com snort[24288]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.135.21:80 -> 192.168.2.4:80

Jun 13 08:11:10 nids2.mycorp.com snort[27809]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.130.21:80 -> 192.168.2.4:80
Jun 13 08:11:10 nids2.mycorp.com snort[27809]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.135.21:80 -> 192.168.2.4:80

Jun 28 08:41:00 nids2.mycorp.com snort[2056]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.130.21:80 -> 192.168.2.4:80
Jun 28 08:41:00 nids2.mycorp.com snort[2056]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.135.21:80 -> 192.168.2.4:80

Snort logs from **nids2.mycorp.com**

[**] SCAN nmap TCP [**]
06/06-12:38:20.891121 199.197.130.21:80 -> 192.168.2.4:80
TCP TTL:52 TOS:0x0 ID:3897 IpLen:20 DgmLen:40
A* Seq: 0x3FD Ack: 0x0 Win: 0x578 TcpLen: 20

=+

[**] SCAN nmap TCP [**]
06/06-12:38:20.921121 199.197.135.21:80 -> 192.168.2.4:80
TCP TTL:51 TOS:0x0 ID:3900 IpLen:20 DgmLen:40
A* Seq: 0x3FF Ack: 0x0 Win: 0x578 TcpLen: 20

=+

[**] SCAN nmap TCP [**]
06/13-08:11:10.221121 199.197.130.21:80 -> 192.168.2.4:80
TCP TTL:52 TOS:0x0 ID:45533 IpLen:20 DgmLen:40
A* Seq: 0x378 Ack: 0x0 Win: 0x578 TcpLen: 20

=+
[**] SCAN nmap TCP [**]
06/13-08:11:10.261121 199.197.135.21:80 -> 192.168.2.4:80
TCP TTL:51 TOS:0x0 ID:45536 IpLen:20 DgmLen:40
A* Seq: 0x37A Ack: 0x0 Win: 0x578 TcpLen: 20

=+

```
[**] SCAN nmap TCP [**]
06/28-08:41:00.282303 199.197.130.21:80 -> 192.168.2.4:80
TCP TTL:52 TOS:0x0 ID:16918 IpLen:20 DgmLen:40
***A**** Seq: 0x363  Ack: 0x0  Win: 0x578  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] SCAN nmap TCP [**]
06/28-08:41:00.322303 199.197.135.21:80 -> 192.168.2.4:80
TCP TTL:53 TOS:0x0 ID:16921 IpLen:20 DgmLen:40
***A**** Seq: 0x365  Ack: 0x0  Win: 0x578  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Source of Trace

All related traces and log files were gathered from my corporate network. The above traces were gathered from Snort IDS sensors **nids1.mycorp.com** and **nids2.mycorp.com** pictured in the diagram which begins Assignment 2. **nids1.mycorp.com** is intended to capture traffic before it is filtered by **border.mycorp.com** and runs a customized rule set excluding many attacks against services, applications, and systems not present in our environment. This balances Snort's performance with the ability to "see who is knocking," assess the general level of threat at our network edge, and detect new attack patterns which may be filtered by the first firewall. The second IDS sensor, **nids2.mycorp.com** runs a far more sensitive, refined rule set designed to account for firewall rules and acceptable traffic patterns in the public DMZ.

Detect was generated by

This detect was generated by the two sensors running Snort version 1.86, an open source network intrusion detection system. The alerts and logs were generated in response to the following rule entered in `scan.rules`, a rule in the `snortrules-stable.tar.gz`[32] standard distribution:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap TCP"; flags:A; ack:0; reference:arachnids,28;
classtype:attempted-recon; sid:628; rev:1;)
```

The format of the rule tells Snort what action to take based on the criteria of a packet. The rule "header" starts with the action - "alert" in this case - which will send the event to the alert output subsystem, rather than the logging output facility, then manage the alert based on additional configuration directives. Following the action, the protocol type is specified. Also specified in the header are the source address and port, direction of the traffic, and the destination address and port pair. Variables are used when possible and specified globally in Snort's configuration file.

In the case of **nids1.mycorp.org**, the `$HOME_NET` destination variable includes all subnets included in the network diagram. The `$HOME_NET` variable from **nids2.mycorp.com** is limited to the public DMZ subnet address of 192.16.2.0/24.

Following the rule header, the rule options provide additional criteria for packet analysis. The rule is designed to capture and alert on traffic generated by nmap, a widely used open source port scanning tool written by Fyodor available at http://www.insecure.org/nmap. While the tool generates a variety of crafted packets designed to map networks for live hosts, probe hosts for available services, and fingerprint operating systems, this particular rule detects the nmap "ACK scan." This scan employs a TCP packet with the least significant bit in the high-order nibble of the 13th byte offset in the header set to 0x0, with no other flags set - in simple terms, only the ACK flag is set. In order for the packet to trigger the alert rule, the TCP acknowledgement number in the 8th-11th bytes offset must also have a value of zero. This traffic is anomalous and constitutes an event of interest; a normal packet has a non-zero value for the acknowledgement number - the 4-byte acknowledgement number field should contain the next sequence number the sender expects to receive.[33] As the initial SYN packet in session establishment occupies one sequence number, a packet with only the ACK flag should never have an acknowledgment number with a zero value. The "ack" field in the Snort rule options is present uniquely for detection of this scanning tactic.[34] However, as demonstrated in *Description of the Attack* below, nmap does not seem to set the acknowledgement number to zero any longer.

Both sensors output alerts a the local syslog facility, log packets to Snort's default logging directory by source IP address, and log to a MySQL database visible through an ACID management interface on an out-of-band network (not shown in diagram) inaccessible from any other subnet.

Probability the source address was spoofed

Based on additional information and the conclusion detailed below, I am certain that the source address was not spoofed and that, in fact, this is entirely a false positive. At the time of the initial detect, however; no such certainty was available, and the normal investigation process for such a reconnaissance attempt was required.

Had reconnaissance ultimately been the intent of this packet, as is typical of nmap scanning attacks as described under *Description of the Attack* below*,* the sender or attacker would hope to gain information about the firewall rule set or host availability within our network perimeter, necessitating return traffic and a legitimate source IP address. Moreover, using TTL comparison techniques, we can determine whether the return path to the source address takes the same number of hops as the arriving packet from the trace. Although these values can be altered by the attacker, we can *make an educated guess* about the likelihood of spoofing and potentially narrow the possibilities of originating platforms.[35] In this case, the TTL of the arriving packets are as follows:

Date	NIDS Sensor	199.197.130.21	199.197.135.21
6/6/2002	nids1.mycorp.com	53	52
	nids2.mycorp.com	52	51

6/13/2002	**nids1.mycorp.com**	53	52
	nids2.mycorp.com	52	51
6/17/2002	**nids1.mycorp.com**	53	52
	nids2.mycorp.com	52	51
6/25/2002	**nids1.mycorp.com**	53	54
	nids2.mycorp.com	-	-
6/28/2002	**nids1.mycorp.com**	53	54
	nids2.mycorp.com	52	53

Depending on the IP implementation of the operating system generating the packets, the default initial TTL will vary. A trace from our public DMZ to the source IP addresses from the detects, taken on 06/06/2002, reveals the number of hops required to reach the source:

```
tracert 199.197.130.21

Tracing route to 199.197.130.21 over a maximum of 30 hops

 1   <1 ms    *      <1 ms  border.mycorp.com [192.168.2.1]
 2    2 ms    2 ms    2 ms  192.168.1.1
 3    *       *       *     Request timed out.
 4    *       *       *     Request timed out.
 5    *       *       *     Request timed out.
 6    *       *       *     Request timed out.
 7    *       *       *     Request timed out.
 8    *       *       *     Request timed out.
 9    *       *       *     Request timed out.
10    *       *       *     Request timed out.
11    *       *       *     Request timed out.
12    *       *       *     Request timed out.
13   21 ms   24 ms   24 ms  199.197.130.21

Trace complete.

tracert 199.197.135.21

Tracing route to 199.197.135.21 over a maximum of 30 hops

 1   <1 ms    *      <1 ms  border.mycorp.com [192.168.2.1]
 2    2 ms    2 ms    2 ms  192.168.1.1
 3    *       *       *     Request timed out
 4    *       *       *     Request timed out.
 5    *       *       *     Request timed out.
 6    *       *       *     Request timed out.
 7    *       *       *     Request timed out.
 8    *       *       *     Request timed out.
 9    *       *       *     Request timed out.
10    *       *       *     Request timed out.
11    *       *       *     Request timed out.
12   25 ms   49 ms   49 ms  199.197.135.21

Trace complete.
```

To reach 199.197.130.21 required 13 hops; reaching 199.97.135.21 required 12 hops. By adding these values to the TTLs of the packet captured by **nids2.mycorp.com** on 06/06/2002, we arrive at the estimated initial TTL values:

6/6/2002

	199.197.130.21	199.197.135.21
TTL of captured packet	52	51
Hops to source from target	13	12
Best Guess Initial TTL Value	65	63

Similar results were obtained for each of detect. Both values, accounting for potentially different routes taken by the trace packet and the captured packet, are very close to 64 - the initial TTL for many Unix-variant IP implementations[36,37,38,39]. The likelihood that the source address was spoofed is decreased based on this correlation between arriving TTL and a most likely initial TTL value of 64, as an actual host responds to the trace at the expected distance vector.

Also decreasing the likelihood of the spoof, and increasing the likelihood that this is a false positive, is the ARIN registration information and the fact that the "attacker" is client business partner:

```
Search results for: 199.197.130.21

OrgName:    Corning Incorporated
OrgID:      CORNIN

NetRange:   199.197.128.0 - 199.197.255.255
CIDR:       199.197.128.0/17
NetName:    CORNING-CBLK
NetHandle:  NET-199-197-128-0-1
Parent:     NET-199-0-0-0-0
NetType:    Direct Assignment
NameServer: NS1.CORNING.COM
NameServer: NS2.CORNING.COM
NameServer: NS3.CORNING.COM
NameServer: NS4.CORNING.COM
Comment:
RegDate:    1994-04-20
Updated:    2001-01-29

TechHandle: ZC107-ARIN
TechName:   Corning Incorporated
TechPhone:  +1-607-974-9000
TechEmail:  dnsadmin@corning.com

# ARIN Whois database, last updated 2002-09-17 19:05
# Enter ? for additional hints on searching ARIN's Whois database.

OrgName:    Corning Incorporated
OrgID:      CORNIN

NetRange:   199.197.128.0 - 199.197.255.255
CIDR:       199.197.128.0/17
NetName:    CORNING-CBLK
NetHandle:  NET-199-197-128-0-1
Parent:     NET-199-0-0-0-0
NetType:    Direct Assignment
NameServer: NS1.CORNING.COM
NameServer: NS2.CORNING.COM
NameServer: NS3.CORNING.COM
```

I was left with the conclusion that either someone was scanning our network with

spoofed addresses and *not* wanting to see the results of the scan (because they would not receive any packets in response), or that someone within Corning, Inc's address space was actually scanning us and thought they would avoid discovery, or that I would not call their administrators. More likely however, I determined that the packet was probably not spoofed; but probably not a scan either.

Description of the attack

The ACK scan (-sA flag) option using nmap can be used to map out firewall rule sets and network exposure by determining which ports are filtered, and whether a firewall is truly stateful.[40,41] The scan sends a packet with only the ACK flag set, which should receive a reset (RST bit set in 13th byte offset of the TCP header) packet in response from an unscreened host. Even if the port is open in the firewall, with no established session, the host should send a RST packet according to protocol[42,43]. When launched against a truly stateful firewall, the firewall should drop the packet on the floor, leaving the scanner to timeout and report the port as filtered (or possibly unreachable if ICMP is also blocked and/or nmap was used with the -P0 switch to avoid sending an ICMP echo request before scanning the target list). As this scan only sends a packet with the ACK bit set, it is not designed to tell if a port is open, which would require a three-way handshake - simply whether the firewall blocks the traffic or allows it to pass through.

However, despite the Snort alert and usual measured concern over nmap scans, several things make this detect a much more interesting event.

First, the non-ephemeral source port, and the reflexive source and destination ports of 80 in all of the detects (except two, which will become clear later) are of interest. The source port can be set with the -g flag in nmap, and might allow an attacker to simulate a response to an HTTP connection from an internal client expecting an ACK packet from port 80 (typically this tactic is used with UDP port 53 or TCP port 20 to simulate return DNS or ftp-data traffic).[44] However, no outbound HTTP connection requests should occur from our DMZ that would necessitate a response from port 80 (nor would the client port be 80), which made this immediately interesting, independent of all the other factors. This tactic raised my concerns a little as the sophistication of the "attack" increased, or at least my level of confusion increased.

A second anomaly discovered in these packets is the interleaving IP identification numbers of each packet pair. By "packet pair," I mean each set of captured packets to the same host, at the same approximate time. The 4th and 5th bytes offset of the IP packet header, reported in the snort log as "ID", always increment by a value of 3, but arrive from different source addresses. While I am not sure of the mathematical probability, I highly doubt that two distinct hosts would consistently use the same pattern and nearly identical IP ID values when there are 65,535 possible values - needless to say, it's pretty

unlikely. This brought me to the conclusion that this was one host with multiple interfaces using the same IP stack - or one interface, alternating spoofed addresses, but this idea is addressed previously.

Finally, the most concerning factor about these packets is that **nids2.mycorp.com** logged them at all. While these packets are not of great concern at the network edge, if logged by **nids1.mycorp.com**, but I had - quite falsely - assumed that the perimeter firewall was stateful and should therefore drop these packets. Further reading of Lance Spitzer's paper, cited previously, explains why this packet made it through. It seems that prior to Firewall-1 4.1 SP2, Firewall-1 allows an un-established ACK packet to reach a screened host.[45] In response, the host would then send a RST - giving away not only the presence of the host, but revealing unfiltered firewall ports without an attacker completing a connection that can be logged by TCP Wrappers, Firewall-1, or other such facilities. As follows from this discovery, the perimeter firewall is clearly ***not*** running a version of Firewall-1 later than, or including SP2 (don't worry it's already the first item in the *Defensive Recommendations*). To confirm this theory, I ran two test nmap scans, one from internal network, which must pass through the fully patched and up to date version of Firewall-1 4.1 - **internal.mycorp.com -** and another test from an external network as the Internet would see my corporate network through the older version. Not surprisingly, the results confirm the failure of pre-SP 2 Firewall-1 to block the packet. That is not a revelation, but more interestingly, we will see that the Snort rule itself is the cause of this false positive, and quite possibly a false negative as well.

The results of the scan coming from the corporate LAN show that **internal.mycorp.com** allows nmap's normal connection (full handshake) scan to pass to port 80, but blocks the ACK scan packet, returning the result that the port is "filtered" as seen in both the second nmap result and the Firewall-1 log:

```
#nmap -P0 -p 80 -sT 192.168.3.2

Starting nmap V. 2.54BETA34 ( www.insecure.org/nmap/ )
Interesting ports on webserver.mycorp.com (192.168.3.2):
Port       State       Service
80/tcp     open        http

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds

# nmap -P0 -p 80 -sA 192.168.3.2

Starting nmap V. 2.54BETA34 ( www.insecure.org/nmap/ )
Interesting ports on webserver.mycorp.com (192.168.3.2):
Port       State       Service
80/tcp     filtered    http

Nmap run completed -- 1 IP address (1 host up) scanned in 36 seconds
```

Firewall-1 on **internal.mycorp.com** drops the packet using the following logging format:

ID / Date /Time / Interface /Origin / Type / Action /Dest Port /SourceAddr / DestinationAddr Proto / Rule /Source Port / Information

```
"6983" "16Jul2002" "15:19:16" "int0" "internal.mycorp.com" "log" "drop" "http" "scanner.mycorp.com" "192.168.2.4"
        "tcp" "0" "37739" " reason: unknown established TCP packet"
"6984" "16Jul2002" "15:19:22" "int0" " internal.mycorp.com " "log" "drop" "http" " scanner.mycorp.com "
        "192.168.2.4" "tcp" "0" "37740" " reason: unknown established TCP packet"
"6985" "16Jul2002" "15:19:28" "int0" " internal.mycorp.com " "log" "drop" "http" " scanner.mycorp.com "
        "192.168.2.4" "tcp" "0" "37741" " reason: unknown established TCP packet"
"6986" "16Jul2002" "15:19:34" "int0" " internal.mycorp.com " "log" "drop" "http" " scanner.mycorp.com "
        "192.168.2.4" "tcp" "0" "37742" " reason: unknown established TCP packet"
"6990" "16Jul2002" "15:19:40" "int0" " internal.mycorp.com " "log" "drop" "http" " scanner.mycorp.com "
        "192.168.2.4" "tcp" "0" "37743" " reason: unknown established TCP packet"
"6991" "16Jul2002" "15:19:46" "int0" " internal.mycorp.com " "log" "drop" "http" " scanner.mycorp.com "
        "192.168.2.4" "tcp" "0" "37744" " reason: unknown established TCP packet"
```

By contrast, the nmap ACK scan from the Internet returns the expected - and unfortunate result that the port is unfiltered, as shown in the nmap results and the tcpdump output (with -tvn flags set). It's acceptable that the port is open, as this is a publicly accessible web server, but the fact that the illegal packet scan succeeds in reconnaissance is concerning.

```
# nmap -sA -T Insane -vvv -p 80 -g 80 webserver_public_ip.mycorp.com

Starting nmap V. 2.54BETA34 ( www.insecure.org/nmap )
Host  (webserver_public_ip.mycorp.com) appears to be up ... good.
Initiating ACK Scan against  (webserver_public_ip.mycorp.com)
The ACK Scan took 0 seconds to scan 1 ports.
The 1 scanned port on  (webserver_public_ip.mycorp.com) is: UNfiltered

Nmap run completed -- 1 IP address (1 host up) scanned in 5 seconds
```

Below is the result of a tcpdump filter listening in the DMZ at the same time the above test scan was launched:

```
# tcpdump -i eth2 -tnXxvvv 'host giachopeful.internet.com'
tcpdump: WARNING: eth2: no IPv4 address assigned
tcpdump: listening on eth1

giachopeful.internet.com.80 > 192.168.2.4.80: . [tcp sum ok] 2926051331:2926051331(0) ack 543427167 win 1024
(ttl 30, id 46656, len 40)
0x0000   4500 0028 b640 0000 1e06 1ec9 xxxx xxxx       E..(.@......BB.o
0x0010   coa8 02040050 0050 ae68 0003 2064 0a5f       .....P.P.h...d._
0x0020   5010 0400 0a3f 0000 0000 0000 0000           P....?........

192.168.2.4.80 > giachopeful.internet.com.80: R [tcp sum ok] 543427167:543427167(0) win 0 (ttl xx, id 34214, len
40)
0x0000   4500 0028 85a6 0000 xx06 ed62 coa8 0204       E..(.......b....
0x0010   xxxx xxxx 0050 0050 2064 0a5f 2064 0a5f       BB.o.P.P.d._.d._
0x0020   5004 0000 91f3 0000 0301 0017 0000           P.............
```

This clearly shows the source host **giachopeful.internet.com** sending the ACK probe with spoofed source port and the webserver responding with a reset packet. Again, this is not breaking news - and is expected behavior - given that the Firewall-1 version is not really maintaining state. What caught my attention is that the acknowledgement number in this test scan has a non-zero value of 543427167 (0x20640a5f). This discovery means that not only did the Snort rule trigger a false positive in the original detect trace; but that this rule in particular does not really detect the nmap TCP ACK scan - not from this version of nmap (2.54BETA34) at least - resulting in a false negative from my test scan.

Attack mechanism

In this case, there was no *attack* mechanism, but rather an anomaly mechanism. The probes in this detect were eventually determined to be the "normal" behavior of a Radware Linkproof device designed to provide the best route to requests from hosts behind such a device.

In an initial effort to determine the source of the packet, I conducted the following search on Google.com - "nmap tcp scan port 80 ack" and followed the thread on the Incidents.org mailing list archives titled "RE: Anyone else seeing TCP ACKs on port 80? - Not LION Worm."[46] This thread discussed similar Snort detects in the context of a load balancers, but the packets in my detect were different than the signature behavior and patterns in Chris Brenton's discussion.

In attempt to find additional evidence, I reviewed web server logs on **webserver.mycorp.com** for the source CIDR block (199.197.x.x/16) in the detects. The server is Internet Information Server and uses the following logging format (the time on the web server differs from the time on **nids1.mycorp.com** by approximately -2:00 minutes, and differs from **nids2.mycorp.com** by -1:13 minutes):

UTC Time / Source IP Address / Client to Server Method / Client to Server URI-stem / Server to Client Status

#Date: 2002-06-06 01:01:27
#Fields: time c-ip cs-method cs-uri-stem sc-status

16:39:33 199.197.135.1 GET /Default.htm 200
16:39:33 199.197.135.1 GET /images/01.jpg 200
16:39:33 199.197.135.1 GET /images/08.jpg 200
16:39:33 199.197.130.1 GET /images/22.jpg 200

Above we see a host from the same registered network making very normal requests to our default page on this site. Curious, but almost relieved, I sent an email inquiry to the address in the ARIN registration and received the following response:

From: DNSTech [DNSTech@corning.com]
Sent: Tuesday, June 25, 2002 15:14
To: me@mycorp.com
Subject: RE: ACK scans from 199.197.130.21 & 199.197.135.21

Bill,

The traffic you are seeing is not an attack, but rather an effect of Corning's multiple ISP load balancing.

In the United States, Corning uses two ISPs for its main Internet access, UUNet (199.197.130.0/24) and Time-Warner (199.197.135.0/24). In order to load-balance outgoing and incoming traffic, we use intelligent load-balancing appliances from Radware Corporation (http://www.radware.com/). These devices, called Linkproofs, also provide Network Address Translation (NAT) for all outgoing connections as well as Corning's eCommerce servers. The Linkproofs are assigned an IP address of 199.197.130.21 for UUnet traffic, and 199.197.135.21 for Time-Warner traffic. All of Corning's usual connections (web, email, etc.) will come from either 199.197.130.1 or 199.197.135.1.
…
For outgoing connections, the Linkproofs maintain a list of which circuit is best to use for access to specific Internet sites. The sites are those most recently accessed. When a connection is initiated to an Internet address that is not in the Linkproof's internal list, the Linkproof sends several probes (at least a PING and one for the protocol being used) down each ISP's circuit. It uses the results to determine which is the best circuit to use. Barring any ISP outages, that result

also is maintained for a period of time so that subsequent connections can use it.

For incoming connections a similar algorithm is followed, though in this case the result is reflected in the ordering of addresses returned by DNS lookups of Corning addresses (e.g. www.corning.com).

An additional benefit of these devices is the removal of a need for BGP4. This results in savings in hardware (less complex routers are required) and staffing (Corning does not have to staff for BGP4 knowledge).

I realize that in today's Internet environment, there is a constant threat of attack from unknown sources. What your site is reporting is one or more of these traffic/load probes. I hope I have alleviated any fears or suspicions that these probes are malicious in nature. They are not malicious and will only occur when needed for Corning traffic that is directed at your site.

Regards,
Security Administrator
Corning Incorporated[47]

While this is a very cordial reply, as if to add insult to my false positive, **nids2.mycorp.com** logged the alert pair seen in the trace logs on 6/25 at the same time I received his email reply, this time with the LinkProof probing port 25 on our mail gateway to determine the fastest route for his message. With this new information, I correlated the alert with the logs from our mail scanner. GFI MailEssentials uses the following format for the log below (there is a large difference in the timestamp on the email and a -2:11 minute difference between the mail gateway and **nids2.mycorp.com**):

Date / Time / Event Type / Sender / Recipient /Total Size (in bytes) / Subject

"2002-06-25","15:26:35","-","DNSTech@corning.com","me@mycorp.com","7920"," RE: ACK scans from 199.197.130.21 & 199.197.135.21"

The device works in two ways. When an Internet client stimulates the LinkProof with a request to a server behind the device; the LinkProof then finds the best path back to the requester using techniques similar to those in Brenton's discussion above and in the Corning administrator's reply.

When a client behind a LinkProof, typically on a network with multiple Internet points-of-presence (two carriers), makes a request to one of our web servers - as is the case with this detect, the Linkproof device intercepts the clients request and probes the destination over the redundant links to determine the fastest or least-cost route. Based on these probes, using what are clearly anomalous packets designed to traverse firewalls and get the fastest confirmed route to the destination host, the request is then routed over the preferred carrier. This explains the interleaving IP identification numbers discussed previously, the IP stack from the same device, but it uses two interfaces, one on each carrier.

This works in a similar, but inverse, way to Speedera ICMP packets as described by Joe Stewart below, but rather than ICMP echo requests, Radware uses more "tricky" packets.

"The true source of the pings is Speedera.net's "Global Traffic Management" system. It isn't a random or sequential sweep of the net; the pings only occur when you make a DNS lookup request for one of their load-balanced cache customers' websites They then use the latency results of the distributed pings to return the IP address of the cache with the fastest route to you. For example. if you connect to any one of the below nameservers using nslookup, and request the address for 'www.speedera.com', your IDS should instantly pick up pings from several servers at once to your IP address."[48]

Correlations

The events of interest mentioned by Robert Wagner in his incidents.org post mentioned above are very similar in stimulus-response behavior. Similar traffic and explanations are offered by John Benninghoff and Javier Romero in the GIAC Report archives.[49] Also providing a similar report was Lawrence Baldwin in a different GIAC report, although none of these reports or packet traces seem to exhibit the reflexive source and destination reports witnessed in the probes to my webservers.[50]

Evidence of active targeting

Knowing that this is not a malicious event, the "targeting" becomes less threatening, but certainly the anomalous packets were directed at our public web server. Fortunately, the targeted traffic was intended to provide the best response to our Internet content.

Severity

Determining severity is complicated in this case; if I based this on the actual detect, an nmap ACK scan, the lethality would be higher than if I determined severity based on the final outcome of my research. In my calculation, I ultimately decided to use the final outcome in the calculation - a probe by a load balancer which never would have entered our DMZ through truly stateful firewall, using an anomalous packet with good intentions.

Criticality	This server is a public web server hosting non-sensitive data and no proprietary data. Compromise of this host would result in little lost business or operational integrity. However, our marketing programs and public communication to clients depends on this server. As always, DMZ host compromise is very concerning, creating a springboard in the perimeter; but in this case an nmap attack alone could not compromise the host.	3
Lethality	As the "attack" did in fact succeed, this attack was not lethal, but did return to proper reconnaissance. Had this been an actual nmap scan, the lethality would still be minimal as only information is determined. Because this information reveals holes in the firewall, however, the lethality should be increased.	2
System Countermeasures	This system has all recent patches installed, runs no unnecessary services, has excellent anti-virus agents, and is under very close monitoring by additional host-based security agents. As this is a public web server, port 80 must remain open and accessible, so naturally there is some level of exposure to the unknown.	5

Network Countermeasures	Because these detects came unimpeded through the not-so-stateful firewall, there was no *active* network defense against this probe. However, the presence of the NIDS sensors provides at least minimal countermeasures in the ability to detect and remediate the scan. Moreover, as most compromise tactics require a session and full TCP connection, the firewall would block full connections to non-HTTP ports as part of the installed rule set. To this particular attack, however, there was little defense in place.	1
Severity	(Criticality 3 + Lethality 2) - (System Countermeasures 5 + Network Countermeasures 1)	-1

Defensive Recommendations

Although this traffic is innocent in its intent, and is designed to provide low cost, intelligent load balancing, a network configuration which allows this device to function best is also vulnerable to reconnaissance attempts such as the nmap ACK scan. As a result, the CheckPoint Firewall-1 device was immediately slated for a much needed upgrade.

Also of note is the false positive generated by Snort. False positives distract the analyst (as quite clearly shown here) and could potentially result in disguise of a malicious attack or scan. Although removing the ACK scan rule from Snort would prevent this false positive, a sensor would no longer detect the original ACK scan with an acknowledgement number with a zero value. Perhaps other methods can be used, at least in our network, to reduce false positives from LinkProof (other than awareness by the analyst) without introducing the risk of false negatives, such as better implementation or configuration of Snort's stream4 preprocessors, designed to detect port scans which exploit session state problems.

Finally, the time should be synchronized among all network hosts, such as the web server, NIDS sensors, firewalls, and mail relay hosts. Correlating events in this incident was not complicated as the volume of data was limited and easily extracted. In a more complex attack, with thousands of insertion attacks, evasion techniques, and multiple source addresses, "every (milli) second counts" in terms of correlation and analysis. Although this recommendation is not an active defense, it provides an analyst with a great deal of insight.

Multiple Choice Question

Based on the alerts and logs below (and on next page) generated from a standard installation of Snort version 1.8.6, what might indicate that these scans are coming from the same host, despite the different source IP addresses? Do not assume that you are seeing all the packets generated by hosts, only those triggering alerts in Snort.

a) The close correlation in the time stamps, network address and destination IP

address.

b) The incrementing pattern seen in the TCP sequence numbers and IP ID fields.
c) These traces are of packets generated from two different hosts.

Snort alerts from **nids1.mycorp.com**

Jun 6 12:37:32 nids1.mycorp.com snort[31510]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.130.21:80 -> 192.168.2.4:80
Jun 6 12:37:32 nids1.mycorp.com snort[31510]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.135.21:80 -> 192.168.2.4:80

Jun 13 08:10:10 nids1.mycorp.com snort[2025]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.130.21:80 -> 192.168.2.4:80
Jun 13 08:10:10 nids1.mycorp.com snort[2025]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.135.21:80 -> 192.168.2.4:80

Jun 25 15:24:26 nids1.mycorp.com snort[783]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.130.21:80 -> 192.168.3.3:**25**
Jun 25 15:24:26 nids1.mycorp.com snort[783]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.135.21:80 -> 192.168.3.3:**25**

Jun 28 08:39:58 nids1.mycorp.com snort[10281]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.130.21:80 -> 192.168.2.4:80
Jun 28 08:39:58 nids1.mycorp.com snort[10281]: [1:628:1] SCAN nmap TCP [Classification: Attempted Information Leak] [Priority: 2]: <eth1> {TCP} 199.197.135.21:80 -> 192.168.2.4:80

Snort logs from **nids1.mycorp.com**

[**] SCAN nmap TCP [**]
06/06-12:37:32.176220 199.197.130.21:80 -> 192.168.2.4:80
TCP TTL:53 TOS:0x0 ID:3897 IpLen:20 DgmLen:40
A* Seq: 0x3FD Ack: 0x0 Win: 0x578 TcpLen: 20

=+

[**] SCAN nmap TCP [**]
06/06-12:37:32.206220 199.197.135.21:80 -> 192.168.2.4:80
TCP TTL:52 TOS:0x0 ID:3900 IpLen:20 DgmLen:40
A* Seq: 0x3FF Ack: 0x0 Win: 0x578 TcpLen: 20

=+

[**] SCAN nmap TCP [**]
06/13-08:10:10.271059 199.197.130.21:80 -> 192.168.2.4:80
TCP TTL:53 TOS:0x0 ID:45533 IpLen:20 DgmLen:40
A* Seq: 0x378 Ack: 0x0 Win: 0x578 TcpLen: 20

=+

[**] SCAN nmap TCP [**]
06/13-08:10:10.311059 199.197.135.21:80 -> 192.168.2.4:80
TCP TTL:52 TOS:0x0 ID:45536 IpLen:20 DgmLen:40
A* Seq: 0x37A Ack: 0x0 Win: 0x578 TcpLen: 20

=+

[**] SCAN nmap TCP [**]
06/25-15:24:26.890538 199.197.130.21:80 -> 192.168.3.3:**25**
TCP TTL:53 TOS:0x0 ID:41685 IpLen:20 DgmLen:40

```
***A**** Seq: 0x3D3  Ack: 0x0  Win: 0x578  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] SCAN nmap TCP [**]
06/25-15:24:26.920538 199.197.135.21:80 -> 192.16.3.3:25
TCP TTL:54 TOS:0x0 ID:41688 IpLen:20 DgmLen:40
***A**** Seq: 0x3D5  Ack: 0x0  Win: 0x578  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] SCAN nmap TCP [**]
06/28-08:39:58.912724 199.197.130.21:80 -> 192.168.2.4:80
TCP TTL:53 TOS:0x0 ID:16918 IpLen:20 DgmLen:40
***A**** Seq: 0x363  Ack: 0x0  Win: 0x578  TcpLen: 20

[**] SCAN nmap TCP [**]
06/28-08:39:58.952724 199.197.135.21:80 -> 192.168.2.4:80
TCP TTL:54 TOS:0x0 ID:16921 IpLen:20 DgmLen:40
***A**** Seq: 0x365  Ack: 0x0  Win: 0x578  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The correct answer is, b) The incrementing pattern seen in the TCP sequence numbers and IP ID fields.

Detect 2 - VPN Policy Violator - Disabled Split Tunneling Saves the Day

Detect Trace Logs

Checkpoint Firewall-1 on **internal.mycorp.com** logged the following drops using this logging format:

ID / Date /Time / Interface /Origin / Type / Action /Dest Port /SourceAddr / DestinationAddr Proto / Rule /Source Port / Information

"1173" "17Jun2002" "15:40:12" "int0" "internal.mycorp.com" "log" "drop" "4773" "66.24.175.38" "66.69.4.10" "tcp" "0" "**http**" "firewall" " reason: unknown established TCP packet"
"1973" "17Jun2002" "16:24:52" "int0" "172.16.0.1" "log" "drop" "3656" "66.24.175.38" "211.202.8.106" "tcp" "0" "**SQL_1433**" "firewall" " reason: unknown established TCP packet"
"2619" "17Jun2002" "17:01:42" "int0" "172.16.0.1" "log" "drop" "14150" "66.24.175.38" "61.188.179.127" "tcp" "0" "**SQL_1433**" "firewall" " reason: unknown established TCP packet"
"4476" "17Jun2002" "19:35:54" "int0" "172.16.0.1" "log" "drop" "49912" "66.24.175.38" "213.26.98.143" "tcp" "0" "**http**" "firewall" " reason: unknown established TCP packet"

Source of Trace

The trace logs above were logged on the internal interface of the Checkpoint Firewall-1 host **internal.mycorp.com** shown in the diagram at the beginning of Assignment 2. Of particular note is that the source addresses are not from the same subnet used for any internal network.

Detect was generated by

The detect was generated as part of the normal logging facilities on our Checkpoint Firewall-1 hosts. These logs are reviewed regularly throughout each day to detect rule set or policy violations, and to troubleshoot network configuration errors and other connectivity issues. These drops were exported during one of these standard reviews.

Probability the source address was spoofed

I almost missed these packets in the standard review of the firewall logs, as the source addresses were from external sources and they were dropped in the firewall, something I see all the time from **border.mycorp.com** . Because home cable modem networks in our area use these addresses, I am almost too used to seeing 66.24.0.0/16 source addresses port scanning our public IP block. However, the filter I had in place at the time to show all packets with a source port of 1433 nearly made me spill my coffee, as no packets (accepted or dropped) should leave our network from this source port. Given the threat levels to Microsoft SQL at the time, this filter was key to ensuring we were not infected. The next bell went off when I noticed that neither the source, nor the destination addresses belonged to any part of our network. Having now captured my full attention, I first wondered if this was an attempt at exploiting some crafty strict-source routing attack against another party, incorrectly using our network as a hop. I dismissed this immediately as the perimeter router does not allow such attacks, nor would it route these

addresses into the internal network.

On closer examination, I was further alarmed when I noticed that the internal firewall had logged these packets. I surmised initially that the there was a problem with our perimeter firewall that was letting in atypical and very dangerous packets. I dismissed this notion because there is no route to the destination addresses via the internal network; the packets would simply be ignored by the perimeter router and **border.mycorp.com**. Moreover, I am not sure what an attacker would gain by scanning an address not in our address range, by traversing our network - if that were somehow possible. My first, and untrained, inclination was to immediately assume that these addresses were spoofed by an inside host and directed to targets outside the corporate boundaries. However, tracing back to the host for conducting passive fingerprinting would not work and do little good as there are no routes internally between these networks required to gather a hop count or trace. My last possibility was that someone had a very poorly configured client workstation on the internal network.

Before determining if the addresses were spoofed, I needed to determine the stimulus and response and how this was even possible in terms of routing. Ultimately, the addresses were not spoofed, as we see below, and the packets are a result of a VPN user with both an improperly configured connection (according to procedure) and a lack of awareness about VPN safety and guidelines - probably my fault as the resident security analyst and trainer.

Description of the attack

In the first packet dropped by Firewall-1, 66.24.175.38 (**xxx-66-24-175-38.xxx.rr.com**) attempts to send a response to packet to what appears to be a client port of 4733 at 66.69.4.10 (**cs66694-10.satx.rr.com**) from source port 80. At this point, I am wondering how this is even possible on our internal network, and I am thankful that the firewall is dropping the packets. Although I cannot be certain without higher fidelity logs with which to correlate, the host at 66.69.4.10 most likely sent a packet to **xxx-66-24-175-38.xxx.rr.com** with the SYN flag set in an attempt to probe for, or connect to, an open web server and **xxx-66-24-175-38.xxx.rr.com** is responding to stimuli. Whether the response contains a SYN/ACK or a RST is uncertain from the Checkpoint logs as the entire TCP packet header is not logged, so I cannot determine if the ports are actually open on the host **xxx-66-24-175-38.xxx.rr.com**. However, the information field in the logs reading "unknown *established* TCP packet" leads me to believe at least the ACK bit was set in the reply, so perhaps this target does have an open web server.

The apparent responses to stimuli continue from **xxx-66-24-175-38.xxx.rr.com** to additional hosts.[51] While the time intervals appear random, all fall within a four hour period.

Response from port 1433 to 211.202.8.106 -
OrgName: Asia Pacific Network Information Centre
CIDR: 210.0.0.0/7

Response from 1433 to 61.188.179.127
OrgName: Asia Pacific Network Information Centre
CIDR: 61.0.0.0/8

These are most likely responses to attempted connections to Microsoft SQL Server's default TCP/IP port, as 1433 is the default service port and scans against this port were running constantly in the Internet during this time (see *Correlations* below). Similar to the HTTP port scan, the host at **xxx-66-24-175-38.xxx.rr.com** was most likely sent a packet with the SYN flag set to the destination port 1433 and is responding to this stimulus. Again, we cannot be certain whether the second part of the TCP session establishment was returned from **xxx-66-24-175-38.xxx.rr.com,** or whether a RST was sent, in any case, fortunately for the victim the packet was dropped at the firewall.

Finally, another packet with a port 80 source address:

Response from port 80 to 213.26.98.143
OrgName: RIPE Network Coordination Centre
CIDR: 213.0.0.0/8

All the probes come from foreign addresses, and as our business has no clients in either Asia or Europe, we can presume that this is not benign traffic.

How was the traffic getting into our LAN in the first place, as we do not share address space with any of the source or destination hosts in the logs? Because we have remote users on the Time-Warner Cable Road Runner service that VPN into the corporate network, and the source address in all packets is that of a RoadRunner client, I searched the remote access and security logs to correlate login times with the times of the probes.

Event Type: Information
Event Source: Router
Event Category: None
Event ID: 20142
Date: 6/17/2002
Time: 13:39:42
User: N/A
Computer: vpnserver.mycorp.com
Description:
The user mycorp.com\vpn_user has connected and has been successfully authenticated on port VPNx. Data sent and received over this link is strongly encrypted.

Event Type: Information
Event Source: Router
Event Category: None
Event ID: 20048
Date: 6/17/2002
Time: 20:40:46
User: N/A
Computer:
Description:
The user mycompany.com\vpn_user connected on port VPNx on 06/17/2002 at 01:39pm and disconnected on 06/17/2002 at 08:40pm. The user was active for 421 minutes 4 seconds. 3635468 bytes were sent and 7147776 bytes were received. The port speed was 100000000. The reason for disconnecting was user request.

When asked, this user was connected to the corporate LAN using VPN over RoadRunner

and did not have a personal firewall in place or running as is required by corporate security guidelines. Fortunately, the Windows 2000 VPN connection settings specified that the host use the VPN connection as the default gateway, rather than the LAN connection to their Road Runner gateway. This is the opposite of the concept know as *split tunneling* and in this case proved to be a crucial defense, given the lack of a personal firewall. The VPN users connect over the Internet to **vpnserver.mycorp.com** and are assigned an IP address from that server with access the Corporate LAN via the tunnel. When a probe from an Internet source hits the client machine via the public Road Runner interface, the response is directed out the VPN gateway's assigned IP address (as the setting specifies), through the tunnel to the LAN, but retains the Internet address assigned by the cable company to the VPN user. As is the case here, these source addresses are dropped by the internal Checkpoint interface.[52]

Had split tunneling been enabled (had the client used RoadRunner for the default gateway, not the VPN connection), any probe of the remote computer would have been successful without a firewall for the VPN client. The probes to tcp port 1433 on the VPN user are particularly terrifying as this user happened to be a programmer with MS SQL running on his VPN client computer. The computer was well patched, and a strong "sa" password was assigned, but the results could have been devastating. Additionally, if the user had not been connected to the corporate LAN via the VPN tunnel, the default gateway would have been the local RoadRunner connection. I have little doubt that this computer was scanned when not connected to the corporate, but escaped compromise only due to frequent patching.

Attack mechanism

Although impossible to determine without better logs, we can make educated guesses about the attack mechanisms based on the time of year. The VPN client did have a web server and had Microsoft SQL Server 2000 installed. While there are no packet traces to confirm to presence of Code Red, Nimda, other more recent port 80 attacks, or the SQL attacks widely discussed, and summarized in CERT/CC Incident Note IN-2002-04 (such as Spida, SQLsnake, and Digispid), the correlation below makes these likely suspects. I am not racing to conclusions without first reviewing some hex, but I don't see any legitimate reasons for hosts from Asia or Europe to connect to SQL Servers on a home user's computer.[53]

The attack would could have been devastating if the VPN user's system were not properly secured. If these were in fact Spida scans, and the victim's computer were compromised, his computer could be used as a launch pad for encrypted attacks into our network. At a minimum, the worm would have attempted to spread itself into the corporate LAN.

Correlations

The 1433 port scans dramatically increased throughout the Internet following the full

disclosure of the open "sa" password vulnerabilities and dissemination of the corresponding attack techniques designed to exploit this, and other, vulnerabilities[54,55]. From **nids1.mycorp.com** alone, between May 20th, 2002 and Aug 31st, 2002. I captured 4373 unique scans to port 1433 from the 211.0.0.0/8 CIDR block using Snort's portscan preprocessor. Analysis on port 1433 scans from the this IP range was conducted using command line tools, such as grep to pass all entries in Snort's portscan.log on **nids1.mycorp.com** with 1433 as the destination port to a text file, then filtering those again for "211.". After the data was checked for validity, I ran a `wc -l` on the file to determine the number of scans. This only accounts for scans to 1433 which attempt connections to more than 4 hosts in under 3 seconds, which is the thresholds passed to Snort's portscan preprocessor. The actual number is potentially higher, and the number of scans to port 1433 from all sources was 24074 for the same period as above.

Between August 18th and August 31st the Internet Storm Center at www.dshield.org reported between 7-23% of the total logged activity as having port 1433 as a source or destination port.[56]

Evidence of active targeting

Again, not assuming anything, but with the correlation of the source network addresses and the volume of scans, I can safely classify this as an automated scan, rather than an attack with keen awareness of the victim. Had the attack succeeded, we may well have seen more targeted packets and traffic between an attacking host and the victim VPN client machine. To the extent that port 1433 is the specific target port, as well as the port 80 probes, there is a small degree of specificity in the attack, even without a unique, isolated attempt to compromise our employee's home computer.

Severity

Criticality	This target host is a VPN client owned by a home user. While compromise of this system itself is inconvenient to his workflow, it would not in itself compromise the corporate LAN. However, because this system has an encrypted tunnel to the LAN and perhaps there is corporate data on his home computer, the level of criticality is increased.	2
Lethality	If the attack had succeeded, and assuming the probes were worms attempting to compromise SQL server, the game is over for this host as the attacker could execute commands using SQL privileges - probably system level permissions.	5
System Countermeasures	This system had all recent patches installed and strong "sa" account password. Unnecessary services were accessible to the Internet (I see this as a system problem since the employee does not need to make SQL Server available to the Internet) and should have been disabled. Additionally, a web server is also running which could have made this host a great launch pad. Despite the patches, the system configuration is risky	2

Network Countermeasures	The fact that split tunneling was disabled, and all packets were sent to and blocked by the corporate firewall, was the only real network countermeasure in place. Had the compromise gone beyond the VPN client to a host in the LAN, the countermeasures would have increased, as we employ firewalls and NIDS at the network edge.	2
Severity	(Criticality 3 + Lethality 2) - (System Countermeasures 5 + Network Countermeasures 1)	1

Defensive Recommendations

Clearly, routing filters and access lists needed to be configured on the remote access VPN server. For no reason should this server ever route a packet without a source address assigned from the DHCP pool reserved for remote clients.

A NIDS sensor on the internal network, while expensive in terms of performance and signature set tuning could at least be run with a minimal signature set to alert on illegal addressing, or even reset sessions and attempted sessions with illegal IP source addresses. As all internal traffic is switched, such a sensor would need to meet considerable performance requirements and have proper placement and visibility to the LAN. Alternately, placing the VPN server and a sensor with no IP address together on the same hub, but sharing a dedicated switch port (like a little bonsia tree sticking out of the switch) would allow the sensor to at least see the decrypted VPN traffic as it enters the LAN. The performance hit on the VPN server is mitigated as no other addressable hosts share that collision domain. This dedicated sensor could maintain a refined, stricter rule set and even employ Snort's FlexResponse feature (requiring an address, however); and as the traffic is generated only by employees, the impact of incorrectly knocking down connections is mitigated.

While the placement of a VPN concentration server inside a corporate LAN is an option, better placement of the VPN and remote access services would add an additional layer of security to the corporate LAN as the traffic could be decrypted before passing through the IDS sensor(s) at the network edge and the corporate firewalls.[57]

The non-technical solution to this problem was raising awareness of more recent changes to remote access security guidelines and requirements. The employee was informed that he needed a personal hardware firewall supplied by the company to sit between his remote office computer and the unscreened Time Warner network. With a firewall in place, the risk of split tunneling is greatly mitigated, as little unsolicited traffic can enter the user's remote LAN. This improves performance for the user if split tunneling is used, but still allows them access to the corporate LAN. If required, additional connection settings or proxy access through the corporate LAN can still be employed, by disabling split tunneling and forcing all connections through the VPN gateway when connected.

Finally, an enforceable requirement that all users working from home do so using a company issued computer with a client certificate, standard system image, and centralized patch configuration for the host would allow more granular management of connected clients.

Multiple Choice Question

Based on the diagram shown at the beginning of Assigment 2, and assuming that all pictured firewalls and routers are properly configured with egress and ingress filters, what could allow the condition where the host addresses shown in the CheckPoint FW-1 logs below are seen in the logs from the internal firewall, generated on the internal interface. The host addresses are neither in use, nor routable on the internal network?

a) Address spoofing from an internal user.
b) Improperly configured VPN clients with split tunneling disabled.
c) An improperly configured external firewall, or connection to an external network not shown on the diagram is allowing the packets through.

ID / Date /Time / Interface /Origin / Type / Action /Dest Port /SourceAddr / DestinationAddr Proto / Rule /Source Port / Information

"1173" "17Jun2002" "15:40:12" "int0" "internal.mycorp.com" "log" "drop" "4773" "66.24.175.38" "66.69.4.10" "tcp" "0" "**http**" "firewall" " reason: unknown established TCP packet"
"1973" "17Jun2002" "16:24:52" "int0" "172.16.0.1" "log" "drop" "3656" "66.24.175.38" "211.202.8.106" "tcp" "0" "**SQL_1433**" "firewall" " reason: unknown established TCP packet"
"2619" "17Jun2002" "17:01:42" "int0" "172.16.0.1" "log" "drop" "14150" "66.24.175.38" "61.188.179.127" "tcp" "0" "**SQL_1433**" "firewall" " reason: unknown established TCP packet"
"4476" "17Jun2002" "19:35:54" "int0" "172.16.0.1" "log" "drop" "49912" "66.24.175.38" "213.26.98.143" "tcp" "0" "**http**" "firewall" " reason: unknown established TCP packet"

The correct answer is, b) Improperly configured VPN clients with split tunneling disabled.

Detect 3 - DDoS Shaft SYNflood - Weak, Anomaly or Backscatter?

Detect Trace Logs

Generated by ACID v0.9.6b21 on Sat September 28, 2002 15:12:57

#(1 - 125128) [2002-09-16 19:10:36] [arachNIDS/253] DDOS shaft synflood
IPv4: 129.125.6.238 -> 192.168.2.x
hlen=5 TOS=0 dlen=40 ID=43887 flags=0 offset=0 TTL=15 chksum=35807
TCP: port=80 -> dport: 80 flags=******S* seq=674711609 ack=1980849715 off=5 res=0 win=61774 urp=7003 chksum=10594
Payload: none

#(1 - 125129) [2002-09-16 19:11:36] [arachNIDS/253] DDOS shaft synflood
IPv4: 129.125.6.238 -> 192.168.2.x
hlen=5 TOS=0 dlen=40 ID=38861 flags=0 offset=0 TTL=15 chksum=40834
TCP: port=80 -> dport: 80 flags=******S* seq=674711609 ack=29000125 off=5 res=0 win=42465 urp=7694 chksum=49898
Payload: none

#(1 - 125130) [2002-09-16 19:12:44] [arachNIDS/253] DDOS shaft synflood
IPv4: 129.125.6.238 -> 192.168.2.x
hlen=5 TOS=0 dlen=40 ID=41125 flags=0 offset=0 TTL=15 chksum=38568
TCP: port=80 -> dport: 80 flags=******S* seq=674711609 ack=1636742653 off=5 res=0 win=12403 urp=10290 chksum=39454
Payload: none

#(1 - 125131) [2002-09-16 19:14:04] [arachNIDS/253] DDOS shaft synflood
IPv4: 129.125.6.238 -> 192.168.2.x
hlen=5 TOS=0 dlen=40 ID=55599 flags=0 offset=0 TTL=15 chksum=24091
TCP: port=80 -> dport: 80 flags=******S* seq=674711609 ack=824787806 off=5 res=0 win=15440 urp=38238 chksum=50198
Payload: none

#(1 - 125132) [2002-09-16 19:15:21] [arachNIDS/253] DDOS shaft synflood
IPv4: 129.125.6.238 -> 192.168.2.x
hlen=5 TOS=0 dlen=40 ID=997 flags=0 offset=0 TTL=15 chksum=54631
TCP: port=80 -> dport: 80 flags=******S* seq=674711609 ack=109718580 off=5 res=0 win=6877 urp=4009 chksum=20234
Payload: none

#(1 - 125133) [2002-09-16 19:16:17] [arachNIDS/253] DDOS shaft synflood
IPv4: 129.125.6.238 -> 192.168.2.x
hlen=5 TOS=0 dlen=40 ID=57227 flags=0 offset=0 TTL=15 chksum=63941
TCP: port=80 -> dport: 80 flags=******S* seq=674711609 ack=1240821363 off=5 res=0 win=32658 urp=43410 chksum=51909
Payload: none

The packets continue for each live host in our public address range, with no clear pattern to the IP ID field, no change to the arriving TTL, and no apparent pattern to the acknowledgement or sequence numbers. Corresponding events for the same packets appear in the ACID console and Snort alert logs for **nids2.mycorp.com** which again logged the attack against hosts placed in the public DMZ.

Source of Trace

All related traces and log files were gathered from **nids1.mycorp.com** on the same corporate network as the previous attacks.

Detect was generated by

This detect was generated by the sensor(s) running Snort version 1.86, an open source network intrusion detection system. The ACID alerts and logs forwarded to MySQL and ACID from Snort were generated in response to the following rule in `ddos.rules`, a rule in the `snortrules-stable.tar.gz`[58] standard distribution:

```
alert tcp $HOME_NET any <> $EXTERNAL_NET any (msg:"DDOS shaft synflood"; flags:S; seq: 674711609;
reference:arachnids,253; classtype:attempted-dos; sid:241; rev:2;)
```

To review this rule using the format from Detect 1, the rule header tells Snort to use the alert subsystem to alert and log any traffic, in any direction where $HOME_NET is part of the source or destination IP pair. Any source or destination port would trigger the alert. The rule options specify that the SYN flag in the 13th byte offset must be set, with no other flags set. The signature TCP sequence number in the 4th-7th bytes offset of the TCP header must carry a value of 674711609.

Probability the source address was spoofed

Most DDoS attacks, particulary SYN flood attacks operate and thrive based on lack of an open connection. This not only consumes resources, but also allows the source address to be spoofed without consequence to the attacker. The attack does not require a response from the client as a DDoS attack is not intended to gain privilege or gather information. Given these goals, it is highly likely that the source address is spoofed. However, because Shaft works according to the attacker-handler-agent model (described and cited below), there is little consequence to the actual person launching the attack if, in fact, the real source IP address is used in the attack, and consequently discovered - usually they would just lose that "agent" host as soon as the incident is reported and the compromised, or "owned," agent is taken off-line.

I was unable to complete a trace back to the source address, which doesn't mean it was spoofed (they could be blocking traces upstream), but the IP address returns the following information on SamSpade:

```
whois -h magic bruno.astro.rug.nl
whois -h whois.domain-registry.nl rug.nl

  Rights restricted by copyright. See
  http://www.domain-registry.nl/whois.php

  Domain name:
    rug.nl (first domain)

  Organisation:
    Rijksuniversiteit Groningen
    Landleven 1
    9747 AD GRONINGEN

  Administrative Contact:
    Robert Janz
```

```
   Phone: +31 50 3633402
   E-mail: postmaster@rug.nl

 Technical Contact:
   Robert Janz
   Phone: +31 50 3633402
   E-mail: rugmail@rc.rug.nl

 Registrar:
   SURFnet B.V.
   Radboudkwartier 273
   3511 CK UTRECHT

 Domain Nameservers:
   ns1.rug.nl              129.125.4.6
   ns2.rug.nl              129.125.4.13
   ns1.surfnet.nl           192.87.106.101

 Domain first registered: 01-01-1980
 Record last updated: 24-05-2000
 Record maintained by: NL Domain Registry
```

A trace from RIPE NCC also returned no valid results or live hosts.

```
traceroute to 129.125.6.238 (129.125.6.238), 64 hops max, 40 byte packets
 1  g002.sinrtr.ripe.net (193.0.0.14)  0.236 ms  0.191 ms  0.174 ms
 2  g0027.nikrtr.ripe.net (193.0.0.145)  0.408 ms  0.407 ms  0.372 ms
 3  BR2.Amsterdam1.surf.net (193.148.15.34)  0.456 ms  0.471 ms  0.456 ms
 4  PO12-0.CR2.Amsterdam1.surf.net (145.145.166.5)  0.501 ms  0.406 ms  0.441 ms
 5  PO0-0.AR5.Groningen1.surf.net (145.145.163.18)  4.297 ms  4.227 ms  4.195 ms
 6  rug-router.Customer.surf.net (145.145.2.2)  4.179 ms  4.207 ms  4.150 ms
 7  * * *
 8  * * *
```

Assuming this is a Shaft DDOS attack, and that the attacker would like to preserve as many agents as possible, I would calculate the risk of the address being spoofed as very high.

Description of the attack

Normally, this attack consists of one or more attackers controlling many, many machines infected with a Trojan backdoor allowing the attacker to execute commands on the compromised host. The attacker, or sometimes an intermediate handler, instructs the agent to flood the victim with enough data to either consume all available bandwidth, or consume all resources on the victim host by forcing it to process a seemingly endless stream of packets. The attacker is made anonymous by either employing the agent drones to "do the dirty work" and take the blame, or by further obfuscating the addresses through spoofing[59].

Attack mechanism

The attack mechanism is described with great care and detail by Dietrich, Long, and Dittrich in their December 2000 paper.[60] Essential, an attacker controls any number of handlers, sending control messages using letter-shifting "encryption" upon which actions are taken. Depending on the control message sent, any variety of packet floods are issued

against the target. Accounting for the correlations cited below, and the limited nature of the attack, perhaps Heymen's assessment is correct that the attacker was confused about the number of shaft handlers under his or her control.

Another theory, used to possibly explain the limited number of packets, is the "reflection" concept advanced by Kenneth Brown in response to Heyman's post.[61] For this to be true, however, the packets would need the SYN and ACK bits set in the TCP header, not just the SYN flag. This is also true if we assume that these packets are the result of backscatter. Moreover, unless someone is (not so cleverly) using the signature TCP sequence number in a new style of attack or port scan, the sequence number would be incremented by a count of one if these packets were backscatter.

Correlations

Leigh David Heymen posted very similar behavior on the incidents.org mailing list.[62] In this post 750 "shaft" SYN packets arrived in the course of three days, with a slower rate of transmission than our detect. These packets, at least the ones posted to the list, were all sent to port 22, whereas our detect was on port 80. Also, the source ports are seemingly randomized in Heymen's detect, unlike the reflexive source ports in this detect. The window size is static and the source IP addresses also change. These packets correlate only due to the slow nature and the signature TCP sequence number.

Evidence of active targeting

This is quite possibly the most difficult part of this detect to determine. As the packets all have only the SYN bit set, it seems as though the attack was initiated against our network directly. However, as the packets are directed at the entire public address range, which is certainly possible with DDoS, it is likely that only more critical or vulnerable servers would be targeted. As all hosts on in our public address space do not offer port 80 services, the effect of a full attack would be limited as non-port 80 hosts would simply reset, whereas a more directed attack at only port 80 servers might generate more traffic as the complete TCP handshake would be attempted. It seems that this attack is more of a probe.

Severity

Criticality	This target in this detect was our entire /25 subnet. Had the attack used the traditional Shaft techniques, it would have impacted the Internet point-of-presence for the entire company. While backup plans, circuits, and continuity plans are in place Internet access is highly critical to the efficient operation of our business	5

Lethality	If the attack had succeeded in launching a Distributed Denial of Service attack, this could have consumed all of our available bandwidth - the final blow, so to speak. However, given the quick detection, and if all packets would have been launched with reflexive source ports and/or similar TCP sequence numbers, cooperation with our ISP upstream to enable committed access rate limitations would have been trivial. Even disabling port 80 traffic at the ISP headend would allow us to continue conducting most critical Internet business.	3
System Countermeasures	A massive DDoS attack against our servers would be very difficult to defend. While some critical services sit behind very fast reverse proxy devices, limited bandwidth to the Internet would make most systems unavailable.	1
Network Countermeasures	Our relationship with our provider includes contingencies for DDoS attacks and can be easily implemented based on known signatures such as these. The local perimeter network is not resilient to such an attack at this point, but alternate - albeit, much slower - connectivity is available.	3
Severity	(Criticality 5 + Lethality 3) - (System Countermeasures 1 + Network Countermeasures 3)	4

Defensive Recommendations

In this case, luck seemed to be the best defense - lucky for us that the attacker did not have more handler's under his or her control. Other strategies include verification of rate limiting procedures with your ISP before an attack happens, RFC 1918 address filtering at the network edge, ingress and egress filtering to prevent address spoofing and to prevent your organization from participating in DDoS attacks, and simply understanding how the attacks occur.[63]

Multiple Choice Question

The following logs generated by a Snort NIDS sensor were forward to you from ACID version v0.9.6b21. Remembering that shaft DDoS attacks have a signature TCP sequence number of 674711609, these packets are the extent of those logged for this source address and this attack signature. Which of the following answers best describes this event?

a) The attacker controlling the shaft DDoS network intended to attack your network, but did not have as many handlers as thought under his or her control.
b) This is backscatter from an attack on another network, using your address space to spoof the attack.
c) This event could be part of a larger DDoS attack against many networks, and your sensor is not seeing the entire attack.
d) Both a) and c) could best explain this event.

```
---------------------------------------------------------------------------
#(1 - 125128) [2002-09-16 19:10:36] [arachNIDS/253]  DDOS shaft synflood
IPv4: 129.125.6.238 -> 192.168.2.x
```

```
hlen=5 TOS=0 dlen=40 ID=43887 flags=0 offset=0 TTL=15 chksum=35807
TCP:  port=80 -> dport: 80  flags=******S* seq=674711609 ack=1980849715 off=5 res=0 win=61774 urp=7003
chksum=10594
Payload: none
---------------------------------------------------------------------------
#(1 - 125129) [2002-09-16 19:11:36] [arachNIDS/253]  DDOS shaft synflood
IPv4: 129.125.6.238 -> 192.168.2.x
hlen=5 TOS=0 dlen=40 ID=38861 flags=0 offset=0 TTL=15 chksum=40834
TCP:  port=80 -> dport: 80  flags=******S* seq=674711609 ack=29000125 off=5 res=0 win=42465 urp=7694
chksum=49898
Payload: none
---------------------------------------------------------------------------
#(1 - 125130) [2002-09-16 19:12:44] [arachNIDS/253]  DDOS shaft synflood
IPv4: 129.125.6.238 -> 192.168.2.x
hlen=5 TOS=0 dlen=40 ID=41125 flags=0 offset=0 TTL=15 chksum=38568
TCP:  port=80 -> dport: 80  flags=******S* seq=674711609 ack=1636742653 off=5 res=0 win=12403 urp=10290
chksum=39454
Payload: none
---------------------------------------------------------------------------
#(1 - 125131) [2002-09-16 19:14:04] [arachNIDS/253]  DDOS shaft synflood
IPv4: 129.125.6.238 -> 192.168.2.x
hlen=5 TOS=0 dlen=40 ID=55599 flags=0 offset=0 TTL=15 chksum=24091
TCP:  port=80 -> dport: 80  flags=******S* seq=674711609 ack=824787806 off=5 res=0 win=15440 urp=38238
chksum=50198
Payload: none
---------------------------------------------------------------------------
#(1 - 125132) [2002-09-16 19:15:21] [arachNIDS/253]  DDOS shaft synflood
IPv4: 129.125.6.238 -> 192.168.2.x
hlen=5 TOS=0 dlen=40 ID=997 flags=0 offset=0 TTL=15 chksum=54631
TCP:  port=80 -> dport: 80  flags=******S* seq=674711609 ack=109718580 off=5 res=0 win=6877 urp=4009
chksum=20234
Payload: none
---------------------------------------------------------------------------
#(1 - 125133) [2002-09-16 19:16:17] [arachNIDS/253]  DDOS shaft synflood
IPv4: 129.125.6.238 -> 192.168.2.x
hlen=5 TOS=0 dlen=40 ID=57227 flags=0 offset=0 TTL=15 chksum=63941
TCP:  port=80 -> dport: 80  flags=******S* seq=674711609 ack=1240821363 off=5 res=0 win=32658 urp=43410
chksum=51909
Payload: none
```

The correct answer is, d) Both a) and c) could best explain this event.

Assigment 3 - Analyze This!

Executive Summary

Contracted by the University to analyze five consecutive, typical days of network intrusion detection system log files, the findings contained in this final deliverable emphasize two key areas of concern. Primarily, the hosts and services under active compromise or attack are discussed with the goal of identifying the most critical areas for immediate remediation. As a second field of concern, the vast quantity of irrelevant alerts and scans logged by the intrusion detection system are examined with the objective of increasing the fidelity of the logs, while retaining the ability to detect legitimate attacks. Fundamentally, the report concludes the following -

- Several systems within the University network have been compromised. These compromised systems present a level of risk and exposure unacceptable to best security practices and require immediate remediation. Regardless of the criticality of these hosts, the compromised state presents a risk to external networks and other, perhaps more critical systems.
- The Snort network intrusion detection system generating the University's log files requires proactive management of the signature and configuration files deployed to the sensors. Improving the reporting accuracy by tuning it to the monitored networks will enhance awareness of attack conditions, aid analysts in making distinctions among real and irrelevant alerts, and increase the efficiency of analysis. In total, 2,038,172 alert or scan entries were logged by Snort during the reporting period. Even with automated correlation tools, this quantity makes the task of meaningful analysis substantially more difficult for human analysts.

Qualitatively and quantitatively significant relationships between hosts and network patterns are isolated when necessary to highlight particular security risks or defensive possibilities; as are correlations with external research or results of previous analysis efforts contracted by the University. Finally, defensive recommendations and the methodology used to enhance relational analysis complete the presentation.

Log Files Analyzed

Three sets of log files spanning the period of June 11th, 2002 through June 15th, 2002 were analyzed according to the scope of work agreed upon between the University and this analyst. The files were generated by one or more Snort intrusion detection sensors listening at key points on the University's network. It is unknown whether the logs were generated by more than one sensor and combined, or if they were generated by a single sensor. The alert log files contain entries generated by specific Snort signatures, where the scan log files are generated by Snort's pre-processors. The out-of-spec log files contain Snort traces of packets which do not meet protocol standard or somehow defy the rules of normal network behavior.

Alert files

Size (bytes)	*Filename*

17,253,046	alert.020611.gz
12,937,967	alert.020612.gz
19,537,545	alert.020613.gz
19,349,046	alert.020614.gz
10,439,117	alert.020615.gz

Scan files

Size(bytes)	*Filename*
24,516,007	scans.020611.gz
18,028,165	scans.020612.gz
27,031,631	scans.020613.gz
26,575,222	scans.020614.gz
17,422,461	scans.020615.gz

Out-of-spec files

Size(bytes)	*Filename*
3,407	oos_Jun.11.2002.gz
261	oos_Jun.12.2002.gz
3,558	oos_Jun.13.2002.gz
2,701	oos_Jun.14.2002.gz
839	oos_Jun.15.2002.gz

Prioritized Detects

The first part of this section details the alerts, or alert patterns, which may indicate the presence of compromised or highly vulnerable hosts - such as those hosting Trojans or backdoors. The reporting then identifies the University's hosts which are potentially responsible for attacks against other systems. This is presented early in the report to bring attention to items for timely remediation. The last part of this section details the highest priority false positives or irrelevant alerts which require the immediate attention of those tasked with administration of the University's Snort sensors.

Compromised or Malicious Hosts within University Network

These hosts are potentially under the control of an attacker and contact should be established with the system owners at once to begin assessing damage and outlining remediation strategies. Rebuilding systems from a clean system state and eliminating vulnerabilities before bringing the systems online is paramount.

- *MY.NET.5.83*

This host is potentially infected by the Ramen worm or SubSeven Trojan. Additional detail, including the hex output from Snort's logs would be required to isolate the actual form of compromise. One of several things is possible based on the alerts below: a) The host is infected with the SubSeven Trojan and is being compromised by internal hosts, or b) It is also possible that the host MY.NET.5.83 offers services not on a standard service ports and these hosts coincidentally connected to it using a client port of 27374, which Snort alerts as potential Trojan activity. More recent versions of Snort signatures might allow this traffic to be more carefully isolated, rather than simply alerting on the port. Other hosts in the log files were listed as alerting on this event as well, but only a single packet and most likely on a coincidental client port. These log files however, provide a

more likely scenario worthy of investigation.

```
06/13-01:48:06.236288  [**] Possible trojan server activity [**] MY.NET.5.19:27374 -> MY.NET.5.83:8262
06/13-01:48:06.236358  [**] Possible trojan server activity [**] MY.NET.5.83:8262 -> MY.NET.5.19:27374
06/13-01:48:06.236446  [**] Possible trojan server activity [**] MY.NET.5.19:27374 -> MY.NET.5.83:8262

06/15-12:57:46.456028  [**] Possible trojan server activity [**] MY.NET.5.88:27374 -> MY.NET.5.83:8907
06/15-12:57:46.456096  [**] Possible trojan server activity [**] MY.NET.5.83:8907 -> MY.NET.5.88:27374
06/15-12:57:46.456163  [**] Possible trojan server activity [**] MY.NET.5.88:27374 -> MY.NET.5.83:8907
06/15-12:57:46.457923  [**] Possible trojan server activity [**] MY.NET.5.88:27374 -> MY.NET.5.83:8907
06/15-12:57:46.466941  [**] Possible trojan server activity [**] MY.NET.5.83:8907 -> MY.NET.5.88:27374
06/15-12:57:46.467404  [**] Possible trojan server activity [**] MY.NET.5.88:27374 -> MY.NET.5.83:8907
06/15-12:57:46.479632  [**] Possible trojan server activity [**] MY.NET.5.83:8907 -> MY.NET.5.88:27374
06/15-12:57:46.481586  [**] Possible trojan server activity [**] MY.NET.5.88:27374 -> MY.NET.5.83:8907

06/15-12:57:47.859351  [**] Possible trojan server activity [**] MY.NET.5.88:27374 -> MY.NET.5.83:7938
06/15-12:57:47.859435  [**] Possible trojan server activity [**] MY.NET.5.83:7938 -> MY.NET.5.88:27374
06/15-12:57:47.859509  [**] Possible trojan server activity [**] MY.NET.5.88:27374 -> MY.NET.5.83:7938
06/15-12:57:47.859767  [**] Possible trojan server activity [**] MY.NET.5.88:27374 -> MY.NET.5.83:7938

06/15-14:54:59.561458  [**] Possible trojan server activity [**] MY.NET.70.177:27374 -> MY.NET.5.83:7938
06/15-14:54:59.561559  [**] Possible trojan server activity [**] MY.NET.5.83:7938 -> MY.NET.70.177:27374
06/15-14:54:59.561759  [**] Possible trojan server activity [**] MY.NET.70.177:27374 -> MY.NET.5.83:7938
06/15-14:54:59.561991  [**] Possible trojan server activity [**] MY.NET.70.177:27374 -> MY.NET.5.83:7938
06/15-14:54:59.562417  [**] Possible trojan server activity [**] MY.NET.5.83:7938 -> MY.NET.70.177:27374
```

- *MY.NET.151.90*

This host can be classified as potentially malicious and/or potentially compromised, given the patterns of traffic seen below (perhaps someone experimenting with system attacks). Although some alerts may simply be false positives based on the packet content, the pattern of alerting justifies investigation.

```
06/14-11:09:48.363626  [**] NIMDA - Attempt to execute cmd from campus host [**]
MY.NET.151.90:1075 -> 207.46.235.150:80

06/12-05:51:44.316386  [**] IRC evil - running XDCC [**] MY.NET.151.90:2344 -> 66.28.132.168:6667
06/12-06:01:46.164054  [**] IRC evil - running XDCC [**] MY.NET.151.90:2675 -> 66.62.70.248:6667
06/12-06:41:44.482526  [**] IRC evil - running XDCC [**] MY.NET.151.90:3974 -> 64.246.34.181:6667

06/13-10:09:42.566935  [**] High port 65535 tcp - possible Red Worm - traffic [**] MY.NET.28.2:38976 -
        > MY.NET.151.90:65535
06/13-10:09:42.788594  [**] High port 65535 tcp - possible Red Worm - traffic [**]
        MY.NET.151.90:65535 -> MY.NET.28.2:38976
```

- *12.151.57.37 & MY.NET.88.245* - See Significant Relationship/Link Graphing Technique section for more detail on these hosts.

Attack Participants within University Network

The subnets listed in the table below contain compromised hosts and require the same remediation process outlined above. In addition to the information security risk and exposure associated with system compromise, the following networks contain hosts that are active, unwilling participants in Unicode manipulation attacks against other hosts

running Microsoft Internet Information Server:

Subnet Address	*Number of Attacking Hosts*
MY.NET.153.0	54
MY.NET.150.0	3
MY.NET.151.0	3
MY.NET.152.0	20
MY.NET.88.0	12

The following two hosts also participate in such attacks:

MY.NET.10.89
MY.NET.83.90

Any alerts for *iis_http_decode: IIS Unicode attack detected* with a source address present in the University's address space warrants attention as these hosts are most likely infected with either Code Red or Nimda. The host should be scanned for open ports not authorized or typical of systems in the address spaces noted above. Follow up with a system owner or administrator is required, and serious consideration of the Defensive Recommendations should be conducted.

False Positives and Irrelevant Alerts or Scan

Under normal circumstances, a properly configured IDS sensor will always produce false positives. For instance, signatures designed to alert to the presence of a Trojan, a backdoor service, or DDoS client will often trigger based on the destination port. As part of coincidental TCP behavior, occasionally these signatures alert on the client port of an established, legitimate connection. Similar conditions occur based on content rules. However, there is an important difference between a signature which alerts erroneously and one that alerts according to expected behavior. The University has enabled many signatures without careful consideration of its security policy throughout the IDS deployment process. Moreover, many of the signatures configured on the University's sensor(s) simply report on normal packet or network behavior, such as ICMP traceroute. While not exhaustive, the University must carefully reconsider the implications of alerting on the following signatures, based on incident handling procedures and the security policy (those in bold demand particular consideration):

INFO Inbound GNUTella Connect accept
WEB-IIS Unauthorized IP Access Attempt
ICMP Destination Unreachable (Protocol Unreachable)
ICMP traceroute
INFO Napster Client Data
WEB-MISC 403 Forbidden
WEB-FRONTPAGE _vti_rpc access
INFO FTP anonymous FTP
WEB-IIS _vti_inf access

ICMP Echo Request Windows
ICMP Destination Unreachable (Communication Administratively Prohibited)
INFO Inbound GNUTella Connect
INFO Outbound GNUTella Connect
ICMP Router Selection
ICMP Fragment Reassembly Time Exceeded

The answer to whether these signatures should remain in deployment should be based on the planned response to such alerts. In some cases, the data is useful for correlation and trending, and in other cases it may be purely statistical or anecdotal. However, especially with regard to peer-to-peer file sharing activity, anonymous FTP, or normal ICMP traffic, if the security policy allows a particular behavior, the NIDS should not be tasked with alerting on it; nor should the University's analysts be burdened with sorting through the irrelevant alerts. In many cases, the alerts can be refined and distributed across subnets so more critical hosts are "watched" closely for atypical patterns

"Top Talkers"

In this section, the top ten network conditions (alerts or scans) are analyzed by purely quantitative means. First, alerts with the highest count are identified and briefly explained; followed by the greatest frequency of source IP-destination port pairs in the scan data. In some cases, the alerts or hosts mentioned overlap with analysis elsewhere in the report; the purpose is to simply draw additional attention to these events based on their frequency. This analysis was conducted prior to completion of the *Prioritized Detects* section and proved an essential aid to the analysis process - narrowing the scope of this assessment. Identifying by quantity also isolates potential false positives, irrelevant alerts or scans, and improperly configured hosts.

Quantitative Alert Analysis

Alert Signature	*Quantity*
SMB Name Wildcard	47748
SNMP public access	45846
spp_http_decode: IIS Unicode attack detected	44360
INFO Possible IRC Access	21951
ICMP Echo Request L3retriever Ping	21936
MISC Large UDP Packet	15403
INFO MSN IM Chat data	8083

- *SMB Name Wildcard*

Sample alert log entry:

```
06/11-10:00:43.644109  [**] SMB Name Wildcard [**] MY.NET.152.45:137 -> MY.NET.11.7:137
06/11-10:00:43.645215  [**] SMB Name Wildcard [**] MY.NET.11.7:137 -> MY.NET.152.45:137
```

The default Snort signature logs an alert based on the content of the packet, in addition to the presence of the reflexive ports. On a Microsoft Windows based networks, this reflexive port traffic is normal and is used for host name resolution and file share access. If seen from an external network address space, concern is justified, as the packet could indicate reconnaissance attempts. However, in the log data, no alerts were generated from outside sources. Assuming that all the hosts on the MY.NET.x.x subnets require access to shared resources, these alerts are irrelevant from security perspective. If the subnets (or host addresses) captured in the alert logs have no legitimate reason to issue NetBIOS name queries, additional investigation may be required. Adjusting this signature on the IDS sensor is addressed in the defensive recommendations.

- *SNMP Public Access*

Sample alert log entry:

```
06/11-12:00:03.935638  [**] SNMP public access [**] MY.NET.70.177:1106 -> MY.NET.5.31:161
06/11-12:00:03.941484  [**] SNMP public access [**] MY.NET.70.177:1106 -> MY.NET.5.31:161
06/11-12:00:03.952514  [**] SNMP public access [**] MY.NET.70.177:1106 -> MY.NET.5.31:161
```

Snort's signature logs the alert on the presence of the word "public" in the content of the packet, in addition to the destination port 161. In later versions of Snort, separate rules exist for both TCP and UDP traffic.

This signature and the resulting alerts provide an excellent tool for auditing community strings that have not been changed from their defaults. The remediation tasks associated with this alert are suggested in the Defensive Recommendations section below, but as none of the source IP addresses logged by this signature originate outside the University, the risk posed by default SNMP community strings is slightly mitigated. However, given the size and unmanaged nature of most of the network, a risk of compromise or reconnaissance from internal sources is very high.

- *spp_http_decode: IIS Unicode attack detected*

Sample alert log entry:

```
06/11-09:50:51.464676  [**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.153.169:3840 -> 211.233.28.192:80
06/11-09:50:51.464676  [**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.153.169:3840 -> 211.233.28.192:80
06/11-09:50:51.502972  [**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.153.169:3840 -> 211.233.28.192:80
```

Snort's http_decode preprocessor listens for packets sent to standard http ports (or those configured in `snort.conf` and converts Unicode character strings into their ASCII equivalents for inspection. Due to the popularity and seemingly endless stream of Unicode-based attacks originating from Code Red variants and Nimda infected hosts, this protection is essential for complete packet analysis and should remain enabled to detect new Unicode tactics, even if all systems are patched. When this preprocessor is invoked, an alert is sent regardless of whether the corresponding rule sets for Code Red, Nimda, and others are enabled.

Connecting to the Internet guarantees that packets scanning for hosts vulnerable to these attacks will continue to enter the University's network. Although vendor patches are widely available to prevent the immediate risk of these known attacks, a layered defensive approach is provided in the Defensive Recommendations section.

- *INFO Possible IRC Access*

Sample alert log entry:

```
06/12-00:43:58.558080 [**] INFO Possible IRC Access [**] MY.NET.151.90:4282 -> 66.28.132.168:6667
06/12-00:44:31.371202 [**] INFO Possible IRC Access [**] MY.NET.151.90:2037 -> 64.246.34.181:6667
```

Snort detects this, and many other chat and instant messaging packets, based primarily on the destination port of 6667. Although many vulnerabilities have been released surrounding IRC and IM clients, the activity or access is not necessarily a security breach. As always, the security policy of the site (in this case the University) determines the severity of this alert. In highly controlled network environments, such activities are strictly prohibited; in other settings the focus on intellectual freedom encourages instant message exchange. Regardless, the University should clearly define its policy in this matter, and then pursue actions to mitigate the risk inherent to allowing these applications, such as user education and more restrictive firewall policies.

Security aside, these technologies can impact network performance if overused. The workstation at MY.NET.151.90 is generating a great deal of IRC Traffic. 21923 of the 21951 (99.87 %) alerts for this activity are generated from this host.

- *ICMP Echo Request L3retriever Ping*

Sample alert log entry:

```
06/11-10:00:43.643800  [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.45 -> MY.NET.11.7
06/11-09:52:15.205569  [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.250 -> MY.NET.11.6
06/11-09:52:21.708638  [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.177 -> MY.NET.11.7
```

This traffic is generally caused by one of two conditions. The first is when the network is scanned by L3's Retriever scanner, in which case the goal of the "attack" is reconnaissance. More likely however, is that Microsoft Windows 2000 hosts are present on the network, as the ICMP packet from these hosts shares the same payload and properties. Given the overlap between the 91 source hosts listed with this alert (all internal and in the subnets probably assigned to the University's users), and the 180 hosts listed under the SMB Name Wildcard alerts, the likelihood of overt scanning with L3's tool is unlikely.

- *MISC Large UDP Packet*

Sample alert log entry:

```
06/11-17:35:25.287208  [**] MISC Large UDP Packet [**] 10.16.2.:3973 -> MY.NET.150.209:3238
06/11-17:35:25.642819  [**] MISC Large UDP Packet [**] 10.16.2.:3973 -> MY.NET.150.209:3238
06/11-17:35:26.016947  [**] MISC Large UDP Packet [**] 10.16.2.:3973 -> MY.NET.150.209:3238
```

Snort generates these alerts on UDP datagrams larger than 4000 bytes. Almost universally, these alerts are triggered by download of streaming media or large stored

media (such as *Pulp Fiction* on DVD) from external sources. Most of these alerts are generated by users of the University network downloading either legitimate or pirated media. See the Critical External Addresses section for additional examples.

- *INFO MSN IM Chat data*

Sample alert log entry:

```
06/11-10:02:36.199166  [**] INFO MSN IM Chat data [**] MY.NET.88.146:1100 -> 64.4.12.158:1863
06/11-09:55:01.541995  [**] INFO MSN IM Chat data [**] MY.NET.88.146:1098 -> 64.4.12.158:1863
06/11-10:05:49.122710  [**] INFO MSN IM Chat data [**] MY.NET.88.146:1100 -> 64.4.12.158:1863
06/11-10:06:46.757251  [**] INFO MSN IM Chat data [**] 64.4.12.158:1863 -> MY.NET.88.146:1100
06/11-10:07:02.483385  [**] INFO MSN IM Chat data [**] 64.4.12.158:1863 -> MY.NET.88.146:1100
```

Snort generates these alerts in response to the presence of port 1863 in the packet. The analysis of this traffic is similar to that regarding IRC access. If this activity is acceptable to the University, then the Snort signature set should be refined to not alert on this traffic, or to do so for only sensitive hosts or subnets.

Quantitative Scan Analysis

All scans listed in the table below are UDP scans. The top ten Source IP/Source Port - Destination IP/Destination Port pairs are provided. Notice that only two source hosts generate all the scans logged.

Source IP Address	Source Port	Destination IP Address	Destination Port	Quantity
12.151.57.37	0	MY.NET.88.245	0	15257
12.151.57.37	1795	MY.NET.88.245	1140	4544
12.151.57.37	516	MY.NET.88.245	1588	3778
MY.NET.5.89	1111	MY.NET.200.112	161	2588
MY.NET.5.89	1111	MY.NET.200.218	161	2551
12.151.57.37	2196	MY.NET.88.245	1248	2527
MY.NET.5.89	1111	MY.NET.200.110	161	2443
MY.NET.5.89	1111	MY.NET.200.36	161	2409
MY.NET.5.89	1111	MY.NET.200.210	161	2394
MY.NET.5.89	1111	MY.NET.200.217	161	2380

- *Port 161 UDP Scans (SNMP)*

In addition to six of the top ten relationships above, the host at MY.NET.5.89 is responsible for 494,152 entries as a source host with source port 1111. All of these scans are directed at UDP port 161, constituting 27% of all scans. For the logging period, even a network highly monitored using snmp queries for performance and fault, this number seems excessively high. If this were legitimate traffic, out of all the scan packets generated, some would find open ports and theoretically send information requests and be detected by the standard Snort signatures (as "public" seems to be the string used

throughout the network). This host is either improperly configured, monitoring a very larger number of hosts with SNMP queries, or using a tool such as PROTOS[64] to generate packets in hopes of targeting or testing vulnerable SNMP systems.

- *Port 123 UDP Traffic (NTP)*

Not shown in the table above, the scan logs report 266,607 UDP scans from MY.NET.60.43 with a source port of 123. Clearly this host can be excluded from the port scan preprocessor as this is simply Network Time Protocol data in transit.

Significant Relationship/Link Graphing Technique - 12.151.57.37 & MY.NET.88.245

So much traffic was generated between these two hosts (as seen in the Quantitative Scan Analysis table) on a variety of ports, that additional investigation techniques were required to analyze the relationship. Because the source was external, a lookup on ARIN was conducted in attempt to classify the system or network.

```
Search results for: ! NET-12-151-56-0-1

OrgName:    ATLIGHTSPEED
OrgID:      LSPD

NetRange:   12.151.56.0 - 12.151.63.255
CIDR:       12.151.56.0/21
NetName:    A-LIGHT112-56
NetHandle:  NET-12-151-56-0-1
Parent:     NET-12-0-0-0-1
NetType:    Reallocated
Comment:
RegDate:    2001-10-11
Updated:    2002-08-22

TechHandle: JM2923-ARIN
TechName:   McCoy, Jeff
TechPhone:  +1-720-264-2029
TechEmail:  jmccoy@atlightspeed.com
```

Nothing exceptionally meaningful was determined from revealing the owner of the address space. Visiting the website of www.atlightspeed.com actually redirected the analyst to http://www.fortrust.biz/ a managed services and platform provider of IT services. The company offers consulting and integration services, in addition to hosting and LAN administration.

In processing the data between these two hosts, we see multiple port scans to several ports on the University host MY.NET.88.24 from 12.151.57.37. As the port relationships are discussed using samples, they are revealed on the link graph following the log section.

15257 UDP packets to port 0 on MY.NET.88.245. :

Jun 11 09:50:44 12.151.57.37:0 -> MY.NET.88.245:0 UDP
Jun 11 09:50:48 12.151.57.37:0 -> MY.NET.88.245:0 UDP
Jun 11 09:50:51 12.151.57.37:0 -> MY.NET.88.245:0 UDP
Jun 11 09:50:56 12.151.57.37:0 -> MY.NET.88.245:0 UDP
Jun 11 09:50:58 12.151.57.37:0 -> MY.NET.88.245:0 UDP

4544 UDP Packets to port 1140 on MY.NET.88.245:

Jun 14 07:42:51 12.151.57.37:1795 -> MY.NET.88.245:1140 UDP
Jun 14 07:43:28 12.151.57.37:1795 -> MY.NET.88.245:1140 UDP
Jun 14 07:43:30 12.151.57.37:1795 -> MY.NET.88.245:1140 UDP

3778 UDP Packets to port 1588 on MY.NET.88.245:

Jun 11 14:10:09 12.151.57.37:516 -> MY.NET.88.245:1588 UDP
Jun 11 14:10:11 12.151.57.37:516 -> MY.NET.88.245:1588 UDP
Jun 11 14:10:52 12.151.57.37:516 -> MY.NET.88.245:1588 UDP

2527 UDP packets to port 1248 on MY.NET.88.245:

Jun 13 07:46:41 12.151.57.37:2196 -> MY.NET.88.245:1248 UDP
Jun 13 07:46:54 12.151.57.37:2196 -> MY.NET.88.245:1248 UDP
Jun 13 07:46:57 12.151.57.37:2196 -> MY.NET.88.245:1248 UDP
Jun 13 07:47:01 12.151.57.37:2196 -> MY.NET.88.245:1248 UDP

SnortSnarf reported 6 different signatures where 12.151.57.37 is present as a source.

1 instances of TFTP - Internal UDP connection to external tftp server
1 instances of TFTP - External UDP connection to internal tftp server

06/11-15:27:18.758436 [**] TFTP - Internal UDP connection to external tftp server [**] 12.151.57.37:69 -> MY.NET.88.245:86
06/14-11:52:08.377086 [**] TFTP - External UDP connection to internal tftp server [**] 12.151.57.37:27906 ->
MY.NET.88.245:69

1 instances of Attempted Sun RPC high port access

06/11-15:38:49.171438 [**] Attempted Sun RPC high port access [**] 12.151.57.37:1 -> MY.NET.88.245:32771

5 instances of EXPLOIT NTPDX buffer overflow
06/11-11:21:35.467492 [**] EXPLOIT NTPDX buffer overflow [**] 12.151.57.37:1029 -> MY.NET.88.245:123
06/13-10:19:16.381417 [**] EXPLOIT NTPDX buffer overflow [**] 12.151.57.37:2057 -> MY.NET.88.245:123
06/14-08:05:43.636738 [**] EXPLOIT NTPDX buffer overflow [**] 12.151.57.37:123 -> MY.NET.88.245:123
06/14-08:44:48.036290 [**] EXPLOIT NTPDX buffer overflow [**] 12.151.57.37:123 -> MY.NET.88.245:123
06/14-10:46:33.056678 [**] EXPLOIT NTPDX buffer overflow [**] 12.151.57.37:1109 -> MY.NET.88.245:123

315 instances of High port 65535 udp - possible Red Worm - traffic [65]
06/11-10:10:55.249738 [**] High port 65535 udp - possible Red Worm - traffic [**] 12.151.57.37:65535 ->
MY.NET.88.245:65280
06/11-10:11:40.741014 [**] High port 65535 udp - possible Red Worm - traffic [**] 12.151.57.37:65535 ->
MY.NET.88.245:65532
06/11-10:13:07.869829 [**] High port 65535 udp - possible Red Worm - traffic [**] 12.151.57.37:65535 ->
MY.NET.88.245:65532
06/11-10:15:08.828512 [**] High port 65535 udp - possible Red Worm - traffic [**] 12.151.57.37:65535 ->
MY.NET.88.245:65535
06/11-10:17:09.740839 [**] High port 65535 udp - possible Red Worm - traffic [**] 12.151.57.37:65535 ->
MY.NET.88.245:65280

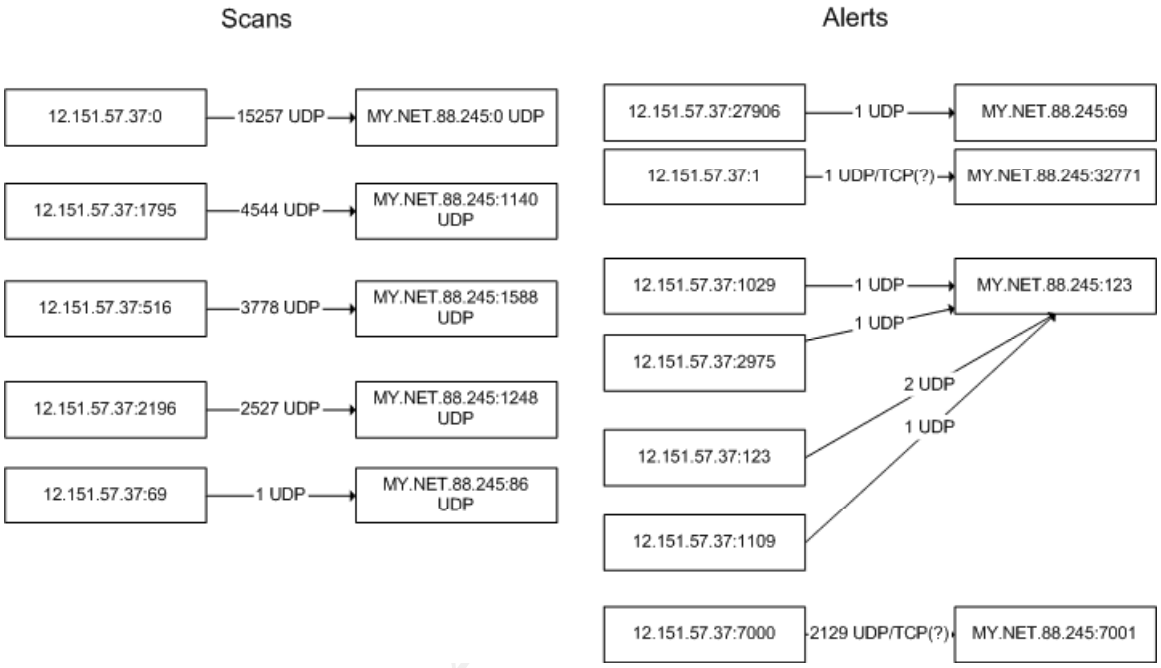
2129 instances of AFS - Off-campus activity
06/11-09:50:37.543001 [**] AFS - Off-campus activity [**] 12.151.57.37:7000 -> MY.NET.88.245:7001
06/11-09:51:05.489616 [**] AFS - Off-campus activity [**] 12.151.57.37:7000 -> MY.NET.88.245:7001

06/11-09:51:26.942677 [**] AFS - Off-campus activity [**] 12.151.57.37:7000 -> MY.NET.88.245:7001
06/11-10:02:20.335881 [**] AFS - Off-campus activity [**] 12.151.57.37:7000 -> MY.NET.88.245:7001

SnortSnarf reported 1 signature where 12.151.57.37 is present as a destination.

87 instances of ICMP Fragment Reassembly Time Exceeded

06/11-11:16:35.362557 [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.88.245 -> 12.151.57.37
06/11-11:16:36.367106 [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.88.245 -> 12.151.57.37
06/11-11:16:41.370621 [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.88.245 -> 12.151.57.37
06/11-11:16:47.380422 [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.88.245 -> 12.151.57.37



The presence of port 7001 and NTP on the same system may indicate the presence of an AFS server. This can be correlated (although not to the same host) with analysis performed by Tod Beardsley in his GCIA practical.[66] However, if in fact the host at 12.151.57.37 is used to provide managed services, the sensor needs be adjusted to limit the number of irrelevant or false positives.

The alerts indicating the presence of a possible Red Worm and a buffer overflow attempt, on a host generating this much traffic between only one other host, certainly warrants investigation to see if the host has definitely been compromised. Contacting the system owner and checking system log files is required for a conclusive statement about this relationship.

Critical External Hosts

This section aggregates several additional external hosts or networks worthy of extraordinary mention. As an output from the previous quantitative analysis, these hosts or networks require additional measures such as reporting to Internet Service Providers and system owners, access control lists or firewall rules dedicated to prevention, or other

measures. A brief justification for "honorable mention" is included with each report.

- *65.92.145.85 - Excessive Code Red or Nimda Attacks*

551 instances of WEB-MISC Attempt to execute cmd
614 instances of spp_http_decode: IIS Unicode attack detected

Registration information retrieved from Dshield.org

65.92.145.85
HostName: HSE-Montreal-ppp337094.sympatico.ca
DShield Profile: Country: CA
Contact E-mail: ip_prov@bellglobal.com
Total Records against IP:
Number of targets:
Date Range: to
Ports Attacked (up to 10): Port Attacks

Fightback: not sent
Whois:
CustName: Nexxia HSE
Address: 87 Ontario Street West Montreal Quebec H2X 1Y8
Country: CA
RegDate: 2001-02-16
Updated: 2001-02-16

NetRange: 65.92.128.0 - 65.92.223.255
CIDR: 65.92.128.0/18, 65.92.192.0/19
NetName: NEXHSE7-CA
NetHandle: NET-65-92-128-0-1
Parent: NET-65-92-0-0-1
NetType: Reassigned
Comment:
RegDate: 2001-02-16
Updated: 2001-02-16

OrgName: Bell Canada
OrgID: LINX

NetRange: 65.92.0.0 - 65.95.255.255
CIDR: 65.92.0.0/14
NetName: BELLNEXXIA-10
NetHandle: NET-65-92-0-0-1
Parent: NET-65-0-0-0-0
NetType: Direct Allocation
NameServer: NS3.BELLGLOBAL.COM
NameServer: NS4.BELLGLOBAL.COM
Comment: ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate: 2001-01-02
Updated: 2002-04-03

TechHandle: PD135-ARIN
TechName: Daoust, Philippe
TechPhone: +1-800-450-7771
TechEmail: noc@in.bell.ca

OrgTechHandle: SYSAD1-ARIN
OrgTechName: Sys Admin
OrgTechPhone: +1-613-785-0886
OrgTechEmail: ip_prov@bellglobal.com

OrgName: Bell Canada
OrgID: LINX

Address: Toronto ON K1G-3J4
Country: CA
Comment:
RegDate: 1990-03-09
Updated: 2002-10-04

AdminHandle: SYSAD-ARIN
AdminName: Sys Admin
AdminPhone: +1-613-785-0886
AdminEmail: ip_prov@bellglobal.com

TechHandle: SYSAD1-ARIN
TechName: Sys Admin
TechPhone: +1-613-785-0886
TechEmail: ip_prov@bellglobal.com

- *202.102.249.118 - Excessive Large UDP packets to MY.NET.88.140:2469*

5810 instances of MISC Large UDP Packet

This IP address, when passed to Google.com was listed on many, many other streaming media sites, primarily http://www.jummpa.com/StreamChane.htm, a site specializing in pirated intellectual property. Clearly this is "normal" traffic for a university (especially the fraternities, if any). However, it constitutes a risk to bandwidth availability and is typical of other IP addresses listed in this alert category. If concerned, the University could block these IP addresses in firewalls or routers, but many other sites/servers exist on any number of given UDP ports.

Registration information retrieved from Dshield.org

HostName: 202.102.249.118
DShield Profile: Country:
Contact E-mail:
Total Records against IP:
Number of targets:
Date Range: to
Ports Attacked (up to 10): Port Attacks

Fightback: not sent
Whois: inetnum: 202.102.249.0 - 202.102.249.255
netname: ZZTB-MIB
country: CN
descr: Zhengzhou Telecom bureau Multimedia Information Bureau,
Zhengzhou city, Henan Provice
450052
admin_c: LZ33-AP
tech_c: LZ33-AP
remarks:
mnt_by: MAINT-CHINANET-HA
changed: zhail@email.online.ha.cn 20010302
status: ALLOCATED PORTABLE
source: APNIC
notify:
mnt_lower:
rev_srv:
start: 3395746048
end: 3395746303
diff: 255

person: Liping Zhong
address: Henan Multimedia Information Bureau
70, Nong Ye Road
ZhengZhou, Henan 450002
CN
country: CN
phone: +86-371-3962276
fax_no: +86-371-3962068
e_mail: antispam@public.zz.ha.cn
nic_hdl: LZ33-AP
mnt_by: MAINT-NULL
changed: zhail@email.online.ha.cn 20001124
source: APNIC
remarks:

- *140.142.8.71- Excessive Large UDP packets to MY.NET.153.159:2259*

5810 instances of MISC Large UDP Packet

This IP address, when passed to GEEKTOOLS.com returned the information below. Most likely another media server, but with potentially a more legitimate purpose (from an academic perspective at least), demonstrating the risk in wholesale blocking or rejection of streaming media packets.

OrgName: University of Washington
OrgID: UWND

NetRange: 140.142.0.0 - 140.142.255.255
CIDR: 140.142.0.0/16
NetName: UW-SEA
NetHandle: NET-140-142-0-0-1
Parent: NET-140-0-0-0-0
NetType: Direct Allocation
NameServer: HANNA.CAC.WASHINGTON.EDU
NameServer: MARGE.CAC.WASHINGTON.EDU
NameServer: NS.UNET.UMN.EDU
Comment:
RegDate: 1990-04-24
Updated: 2000-03-17

Additional information can be obtained from setting a workstation's default name server to that listed above:

>nslookup
Default Server: myisp's.dns.server
Address: 192.168.1.x

> server hanna.cac.washington.edu
Default Server: hanna.cac.washington.edu
Address: 140.142.5.5

> 140.142.8.71
Server: hanna.cac.washington.edu
Address: 140.142.5.5

Name: media-wm-1.cac.washington.edu
Address: 140.142.8.71

Out of Spec Packets

The amount of data in the OOS files was manageable and could be visually correlated to return at least some meaningful results. Primarily, the host at MY.NET.150.209 is running a peer-to-peer file sharing program such as Bearshare or Morpheus or GNUtella. Again, the severity of this knowledge depends on the University's security policy. What is interesting about this detect, however, is the way in which either the application's socket implementation, or the host's TCP/IP stack is corrupting the packets. Frequently, tools like nmap and other OS fingerprinting application will intentionally generate this traffic to see how a host responds. In this case, it seems that the application simply generates several packets with random TCP flags and option set.

```
:06/12-00:39:40.957085 193.6.40.86:55089 -> MY.NET.150.209:6346
 -TCP TTL:48 TOS:0x0 ID:13257  DF
 -21S***** Seq: 0xABA41371   Ack: 0x0   Win: 0x16D0
 -TCP Options => MSS: 1460 SackOK TS: 2629816 0 EOL EOL EOL EOL
--
 -=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 :06/12-00:39:43.957516 193.6.40.86:55089 -> MY.NET.150.209:6346
 -TCP TTL:48 TOS:0x0 ID:13258  DF
 -21S***** Seq: 0xABA41371   Ack: 0x0   Win: 0x16D0
 -TCP Options => MSS: 1460 SackOK TS: 2630116 0 EOL EOL EOL EOL
--
 -=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 :06/12-21:20:30.868375 24.112.58.210:2656 -> MY.NET.150.209:6346
 -TCP TTL:114 TOS:0x0 ID:10611  DF
 -*1SFRPAU Seq: 0x690547   Ack: 0xF2A70AEA   Win: 0x5018
 -22 38 EF 63 00 00 89 1A 46 26 CA E8 23 98 FF AE  "8.c....F&..#...
--
 -=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 :06/12-21:23:55.685430 24.112.58.210:2656 -> MY.NET.150.209:6346
 -TCP TTL:114 TOS:0x0 ID:11922  DF
 -2*SFRPAU Seq: 0x54D2CF3   Ack: 0x6B0AF3   Win: 0x5018
 -TCP Options => EOL EOL
 -=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 :06/12-21:24:24.400833 24.112.58.210:2656 -> MY.NET.150.209:6346
 -TCP TTL:114 TOS:0x0 ID:25238  DF
 -21SF***U Seq: 0xA6054D   Ack: 0xD6210AF4   Win: 0x5018
 -TCP Options => EOL EOL SackOK
--
 -=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 :06/12-21:25:35.910653 24.112.58.210:2656 -> MY.NET.150.209:6346
 -TCP TTL:114 TOS:0x0 ID:34207  DF
 -*1SF**AU Seq: 0x54F   Ack: 0x294B0AF5   Win: 0x5018
 -TCP Options => EOL EOL
```

Another packet seen in the OOS files demonstrates the use of Explicit Congestion Notification, as seen in the use of the reserved TCP flags. Although it is possible to use Quality of Service options on newer platforms, this is unusual to see in a TCP packet from a host, which - given the destination port 80 - is probably generating the traffic, rather than a router that might use ECN.

```
06/11-21:30:35.373910 68.80.114.202:1250 -> MY.NET.5.96:80
TCP TTL:108 TOS:0x0 ID:12297  DF
21SF*P*U Seq: 0x5B3064   Ack: 0x2169   Win: 0x5010
TCP Options => EOL EOL EOL EOL EOL EOL SackOK
```

This may be an attempt at OS fingerprinting, or perhaps the indication of a Trojan port, where unusual TCP options are used to communicate between hosts. Port 1269 was listed

as a possible Trojan port during a Google search.

```
06/13-17:39:58.851407 64.4.124.151:3193 -> MY.NET.88.165:1269
TCP TTL:113 TOS:0x0 ID:52404  DF
21**R**U Seq: 0xBCCA1D8   Ack: 0x7D86   Win: 0x5010
0C 79 04 F5 0B CC A1 D8 00 00 7D 86 04 E4 50 10  .y........}...P.
79 34 A9 3E 00 00 C2 31 19 C0 20 0F B0 1A 62 7A  y4.>...1.. ...bz
F3 93                                            ..
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/13-17:46:22.699466 64.4.124.151:0 -> MY.NET.88.165:3193
TCP TTL:113 TOS:0x0 ID:61990  DF
21**RP*U Seq: 0x4F50D80   Ack: 0x1D87D87   Win: 0x5010
3C EC 50 10 7B 30 F7 71 00 00 3E FA 61 41 AF A4  <.P.{0.q..>.aA..
76 86 A2 1B F5 D2                                v.....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/13-17:54:54.956901 64.4.124.151:4 -> MY.NET.88.165:3193
TCP TTL:113 TOS:0x0 ID:65215  DF
21**R*** Seq: 0x4F50FC7   Ack: 0x31D87D88   Win: 0x5010
TCP Options => EOL EOL EOL EOL EOL EOL SackOK SackOK SackOK EOL Opt 53 Opt 53 Opt 53 Opt 53
        Opt 53 Opt 53 Opt 53 Opt 53 Opt 53 Opt 53 Opt 53 Opt 53 Opt 53 Opt 53 Opt 53 Opt 53 Opt 53
        Opt 53 Opt 53 Opt 53 Opt 53 Opt 53 Opt 53 Opt 53 Opt 53 Opt 53 Opt 53 Opt 53 Opt 53 Opt 53

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/13-18:00:01.789438 64.4.124.151:3193 -> MY.NET.88.165:1269
TCP TTL:113 TOS:0x0 ID:21275  DF
21**R**U Seq: 0x11122   Ack: 0xD1D87D89   Win: 0x5010
TCP Options => EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/13-18:16:02.185414 64.4.124.151:3193 -> MY.NET.88.165:1269
TCP TTL:113 TOS:0x0 ID:1338  DF
21**RP*U Seq: 0x1566B78C   Ack: 0x7D8C   Win: 0x5010
TCP Options => EOL EOL Opt 23 (3): 1FFC Opt 252

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/13-18:18:59.662527 64.4.124.151:3193 -> MY.NET.88.165:1269
TCP TTL:113 TOS:0x0 ID:15215  DF
*1SF**** Seq: 0x163141D8   Ack: 0x7D8D7EAC   Win: 0x5010
0C 79 04 F5 16 31 41 D8 7D 8D 7E AC 00 83 50 10  .y...1A.}.~...P.
78 30 9D E1 00 00 C7 DE 40 04 C4 CE 52 1C DE 7D  x0......@...R..}
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/13-18:36:41.669086 64.4.124.151:3193 -> MY.NET.88.165:1269
TCP TTL:113 TOS:0x0 ID:37288  DF
21**RP*U Seq: 0x1ADFD1D8   Ack: 0x927D90   Win: 0x5010
TCP Options => EOL EOL
```

Based on the presence of such strange packets, an external search for this host was conducted. The results likely lead to a home user, as indicated by the “DSL DHCP Range in the GEEKTOOLS.com output.

```
Ntelos North Cisco DSL DHCP Range #2 CFW-64-4-124-NNC (NET-64-4-124-0-1)
                    64.4.124.0 - 64.4.124.255
Ntelos Inc. NTELO-BLK-2 (NET-64-4-96-0-1)
                    64.4.96.0 - 64.4.127.255
```

Defensive Recommendations

Harden Host and Perimeter Defenses

- The spp_http_decode preprocessor and similar Snort signatures which detect Nimda and the Code Red variants should be used in conjunction with an application layer firewall with integrated content inspection tools designed to detect and block malware and attacks similar to the Unicode-style attacks witnessed on the University's network. This strategy reduces the introduction of malicious http code into the environment (by blocking it at the network edge), and enhances the fidelity of Snort's logs by reducing irrelevant alerts.

- Ensure that hosts on the network are updated with a quality virus engine and loaded with the latest virus signatures.

Limit False Positives & Irrelevant Alerts

- Adjust IDS sensor to limit `SMB Name Wildcard` alerts. This is normal traffic and provides only informative detail. As mentioned previously, if subnets in the alert data have no legitimate need to host Windows file shares, and sensors can be placed per subnet or VLAN, then this signature may remain in place, although highly refined in its placement on the network.

- The signatures generating the `SNMP Public Access` can remain in place if a concerted effort is made to change the default community strings throughout the University's network. If no effort is to be made, then nothing is gained from capturing this traffic. As the effort is made to change the strings, a corresponding decrease in the number of alerts should occur, allowing systems administrators to identify systems still using the default strings.

- The following line should be added to the `snort.conf` file on the sensor(s) to reduce the amount of irrelevant entries in the scan log files generated by normal NTP traffic:

```
preprocessor portscan-ignorehosts: MY.NET.60.43
```

- Consider implementation of a statistical packet anomaly based intrusion detection system to run in conjunction with Snort as another layer of analysis. As these systems also produce a great deal of detail regarding network traffic patterns, resources above those currently allocated need to be provisioned for the monitoring of such a system. While valuable, a system such as this must be closely managed; if the University takes a more proactive role in managing Snort's irrelevant alerts, then this is a logical next step in implementing a layered detection approach.

Upgrade Snort Binaries

As of this writing Snort version 1.9.0 has officially been released. It is highly recommended that the University test this system in a lab environment, and assuming the tests are successful, schedule the release for turnover into the production network.

Features such as "flow" and revisions to not only the signature set, but critical preprocessors as well would enhance the usefulness of Snort as the IDS of choice for the University.

Methodology

For the alert log files, each file was downloaded to a common directory, renamed with a .txt extension, and used concatenated with the DOS "copy" command into a single file, large file. This was done to gain a holistic perspective on the alerts. The file was processed using SnortSnarf.pl[67] to sort the files in a well-organized HTML format. Using Apache installed on my Windows XP laptop, I reviewed the SnortSnarf output.

```
c:\>sans3\alerts>copy alert020611.txt+alert020612.txt+alert020613.txt+alert020614.txt+alert020615.txt alerts
alert020611.txt
alert020612.txt
alert020613.txt
alert020614.txt
alert020615.txt
      1 file(s) copied.
```

I attempted to use the SnortSnarf.pl script on the scan files as well; both on each separate file, and on a file concatenated in the same manner as the alert files, but could not effectively parse all of the files - even on a very powerful server. As an alternative, I placed the scan files in a unique directory and used the following sed expression (text is wrapped from a single line) to convert them into comma separated value (.csv) files:

```
$ sed 's/\(Jun [0-9]\{2\} [0-9]\{2\}:[0-9]\{2\}:[0-9]\{2\}\)
\([^.]*.[^.]*.[^.]*.[^.]*\)\:\([0-9]\{1,5\}\) \(->\) \([^.]*.[^.]*.[^.]*.[^.]*\)\:\([0-
9]\{1,5\}\) \(.*\)/\1\,\2\,\3\,\5\,\6\,\7/' scans.020612 > scans.020612.csv
```

The expression turns this:

```
Jun 11 09:50:01 205.188.228.33:15050 -> MY.NET.151.85:6970 UDP
Jun 11 09:50:00 MY.NET.153.126:2176 -> 61.145.113.82:80 SYN ******S*
```

into this:

```
Jun 11 09:50:01,205.188.228.33,15050,MY.NET.151.85,6970,UDP
Jun 11 09:50:00,MY.NET.153.126,2176,61.145.113.82,80,SYN ******S*
```

Due to the large size of the files, and given that most of the powerful servers in our organization are Windows-based, I installed Cygwin and ran the `sed` script within the Cygwin shell. I then placed the imported the files into a Microsoft SQL database by creating DTS package which imports the file and creates a table to store the data. This technique leverages the graphical tools present in Microsoft SQL Enterprise Manager and *very* quickly imports the data; all that is required is that the data source is in an easily parsed format, in this case `.cvs`. The columns are simply transformed from the text-only data source into table columns within the database using the CopyColumns transformation built into SQL. The table can be created with the following script; the data needs to be imported using DTS:

```
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[scans2]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[scans2]
GO

CREATE TABLE [dbo].[scans2] (
        [ID] [int] IDENTITY (1, 1) NOT NULL ,
        [Date] [datetime] NULL ,
        [SourceIP] [varchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
        [SourcePort] [varchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
        [DstIP] [varchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
        [DstPort] [varchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
        [Attack] [varchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO
```

The following SQL queries were used to analyze the scan data[68]:

scans_SourceIP_Count.sql

```
SELECT    SourceIP, COUNT(*) AS COUNT
FROM      dbo.scans2
GROUP BY SourceIP
ORDER BY COUNT DESC
```

scans_SourceIP_DstPort_Count.sql

```
SELECT    SourceIP, DstPort, COUNT(*) AS COUNT
FROM      dbo.scans2
GROUP BY SourceIP, DstPort
ORDER BY COUNT DESC
```

scans_SourceIP_DstIP_DstPort_Count.sql

```
SELECT    SourceIP, DstIP, DstPort, COUNT(*) AS COUNT
FROM      dbo.scans2
GROUP BY SourceIP, DstIP, DstPort
ORDER BY COUNT DESC
```

scans_SourceIP_SourcePort_DstIP_DstPort_Count.sql

```
SELECT    SourceIP, SourcePort, DstIP, DstPort, COUNT(*) AS COUNT
FROM      dbo.scans2
GROUP BY SourceIP, SourcePort, DstIP, DstPort
ORDER BY COUNT DESC
```

scans_SourceIP_SourcePort_DstIP_DstPort_Attack_Count.sql

```
SELECT    SourceIP, SourcePort, DstIP, DstPort, Attack, COUNT(*) AS COUNT
FROM      dbo.scans2
GROUP BY SourceIP, SourcePort, DstIP, DstPort, Attack
ORDER BY COUNT DESC
```

scans_per_hour_of_day.sql[69]

```
SELECT CONVERT(varchar(8),Date,1) AS 'Day',
    SUM(CASE WHEN DATEPART(hour,Date) = 0 THEN 1 ELSE 0 END) AS '00:00:00-00:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 1 THEN 1 ELSE 0 END) AS '01:00:00-01:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 2 THEN 1 ELSE 0 END) AS '02:00:00-02:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 3 THEN 1 ELSE 0 END) AS '03:00:00-03:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 4 THEN 1 ELSE 0 END) AS '04:00:00-04:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 5 THEN 1 ELSE 0 END) AS '05:00:00-05:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 6 THEN 1 ELSE 0 END) AS '06:00:00-06:59:59',
```

```
        SUM(CASE WHEN DATEPART(hour,Date) = 7 THEN 1 ELSE 0 END) AS
'07:00:00-07:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 8 THEN 1 ELSE 0 END) AS '08:00:00-08:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 9 THEN 1 ELSE 0 END) AS '09:00:00-09:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 10 THEN 1 ELSE 0 END) AS '10:00:00-10:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 11 THEN 1 ELSE 0 END) AS '11:00:00-11:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 12 THEN 1 ELSE 0 END) AS '12:00:00-12:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 13 THEN 1 ELSE 0 END) AS '13:00:00-13:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 14 THEN 1 ELSE 0 END) AS '14:00:00-14:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 15 THEN 1 ELSE 0 END) AS '15:00:00-15:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 16 THEN 1 ELSE 0 END) AS '16:00:00-16:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 17 THEN 1 ELSE 0 END) AS '17:00:00-17:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 18 THEN 1 ELSE 0 END) AS '18:00:00-18:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 19 THEN 1 ELSE 0 END) AS '19:00:00-19:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 20 THEN 1 ELSE 0 END) AS '20:00:00-20:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 21 THEN 1 ELSE 0 END) AS '21:00:00-21:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 22 THEN 1 ELSE 0 END) AS '22:00:00-22:59:59',
    SUM(CASE WHEN DATEPART(hour,Date) = 23 THEN 1 ELSE 0 END) AS '23:00:00-23:59:59'
FROM  Scans2
GROUP BY CONVERT(varchar(8),Date,1)
ORDER BY CONVERT(varchar(8),Date,1)
```

---

[1] Northcutt, S. et. al. "Chapter 17 - Maintaining a Security Perimeter" Inside Network Perimeter Security: The Definitive Guide to Firewalls, VPNs, Routers, and Intrusion Detection Systems. New Riders:Indianapolis, IN. 2003. p. 483-491

[2] Kim, Gene & Robson, Chris. "Sudden Production Environments: Confessions and Atonments of an R&D Manager." Tripwire, Inc. 23 August 2002. URL: http://www.tripwire.com/events/archived_seminars/index.cfm? (23 August 2002).

[3] Kim, G. & Robson, C. ibid.

[4] Kim, G. & Robson, C. ibid. During slide 14 ot the presentation, Robson, quite ironically when viewed from the perspective of Snort management, uses the phrase "Throwing the pig over the wall" to describe poorly defined acceptance, hand-off, and turnover procedures.

[5] Green, Chris. "Re: [Snort-sigs] Autoupdate snort signature" Snort Users Listserv. 4 October 2002. Archive URL: http://marc.theaimsgroup.com/?l=snort-sigs&m=103372946400999&w=2 (4 October 2002)

[6] While this statement is generally accurate, organization may wish to deploy certain signatures for statistical purposes, and perhaps to detect attacks from inside their own network using exploits against systems not present in their production environment (e.g. an exploit against MS IIS from an Apache-only environment.).

[7] Northcutt, S. et. al. ibid.

[8] Northcutt, S. et. al. ibid.

[9] Kim, G. & Robson, C. ibid.

[10] Northcutt, S. et. al. ibid.

[11] Windows Script Host 5.6. Microsoft Corporation. Download URL: http://download.microsoft.com/download/winscript56/Install/5.6/NT5/EN-US/scripten.exe

[12] OpenSSH Project. URL: http://www.openssh.org. (2 December 2001).

[13] University of Washington Department of Genome Sciences. "OpenSSH Public Key Authentication Setup." http://cfm.gs.washington.edu/security/ssh/client-pkauth/. (5 September 2002).

[14] Byrne, Jason. "SuSE mailinglist: RE: [suse-sparc] startup script for sshd" 8 December 2000. URL: http://lists.suse.com/archive/suse-sparc/2000-Dec/0036.html. (16 May 2002).

[15] Vesperman, Jennifer. "CVS Administration." O'Reilly Network Linux Devcenter. 3 January 2002. URL: http://linux.oreillynet.com/pub/a/linux/2002/01/17/cvsadmin.html. (14 August 2002).

[16] Vesperman, Jennifer. "Introduction to CVS." O'Reilly Network Linux Devcenter. 3 January 2002. URL: http://linux.oreillynet.com/pub/a/linux/2002/01/03/cvs_intro.html. (14August 2002).

[17] Concurrent Versions System. URL: http://www.cvshome.org.

[18] Francis, Andrew. "Setting up a CVS repository on FreeBSD in 5 minutes." 13 April 2002. URL: http://www.sullust.net/docs/quickcvs.html. (3August 2002).

[19] Allbery, Russ. cvslog. URL: http://www.eyrie.org/~eagle/software/cvslog/cvslog.html (3 August 2002).

[20] ViewCVS Project. URL: http://viewcvs.sourceforge.net/. (12 August 2002).

[21] The versions of RPM used in testing were RPM Build 4.0.3 & RPM 4.0.3. These packages are available on any Red Hat mirror site. Different versions may be required for any particular installation of Red Hat.

[22] Blackman, David. "Debian Package Management, Part 2: A Developer's Guide." Linux Journal. 21 June, 2002. URL:http://www.linuxjournal.com/article.php?sid=4610 (1 September 2002).

[23] OpenPKG Project. URL: http://www.openpkg.org/ (1 September 2002).

[24] Bailey, Edward C. "Chapter 9 - The Philosophy Behind RPM." Maximum RPM: Taking the Red Hat Package Manager to the Limit.. Red Hat, Inc.:Durham, NC. 2000. URL: http://www.rpm.org/max-rpm/ch-rpm-philosophy.html (1 September 2002).

[25] Poirier, Dan. "Packaging software with RPM, Part 1; Using RPM on Red Hat Linux 7.1." November 2001. IBM developerWorks. URL: http://www-106.ibm.com/developerworks/library/l-rpm1/ (1 September 2002).

[26] Poirier, Dan. "Packaging software with RPM, Part 2: Building without root, patching software, and distributing RPMs." December 2001. IBM developerWorks. URL: http://www-106.ibm.com/developerworks/library/l-rpm2/ (1 September 2002).

[27] Bailey, Edward C. "Chapter 13 - Inside the Spec file; Macros: Helpful Shorthand for Package Builders." Maximum RPM: Taking the Red Hat Package Manager to the Limit.. Red Hat, Inc.:Durham, NC. 2000. URL: http://www.rpm.org/max-rpm/s1-rpm-inside-macros.html. (1 September 2002).

[28] ipmadmin.vbs. Shinn, Bill (author). Included in practical submission, Bill_Shinn_GCIA.zip.

[29] CVSNT Enhanced CVS Server. URL: http://www.cvsnt.com. Download URL: http://www.cvsnt.org/cvsnt_1.11.1.3.exe (4 August 2002).

[30] Network Simplicity OpenSSH on Windows. URL: http://www.networksimplicity.com/openssh (24 April 2002).

[31] Dell, Jeff. IDS Policy Manager version 1.3 Beta 3. URL: http://www.activeworx.com/downloads/index.htm (2 May 2002)

[32] At the time of this writing, the current Snort stable rule set distribution was available at: http://www.snort.org/dl/rules/snortrules-current.tar.gz

[33] Stevens, W. TCP/IP Illustratred, Volume 1: The Protocols. Addison-Wesley:Boston. 1994. p 226-7.

[34] Green, C. and Roesch, M. "Chapter 2, Section 2.3.15 Ack" Snort Users Manual: Snort Release: 1.9.x. SnortUsersManual.pdf. 26 April 2002. URL: http://www.snort.org/docs/writing_rules/chap2.html#tth_sEc2.3.15 (9 August 2002)

[35] Novak, J. Track 3 - Intrusion Detection In-Depth 3.2, 3.3: Network Traffic Analysis Using tcpdump. SANS Institute. 2002

[36] Northcutt, S., Novak, J., and McLachlan, D. Network Intrusion Detection: An Analyst's Handbook, Second Edition. New Riders:Indianapolis, IN. 2001

[37] Swiss Academic & Research Network, MAP Data Centre. "Default TTL Value in TCP/IP." April 2001. URL: http://www.map2.ethz.ch/ftp-probleme.htm. (24 June 2002).

[38] Honeynet Project. "Knowing your Enemy: Passive Fingerprinting *Identifying remote hosts, without them knowing*." 4 March 2002. URL: http://www.honeynet.org/papers/finger/ (14 June 2002).

[39] Honeynet Project. "List of fingerprints for passive fingerprint monitoring." 23 May 2000. URL: http://www.honeynet.org/papers/finger/traces.txt. (14 June 2002).

[40] Spitzner, L. "Auditing Your Firewall Setup". 12 December 2000. URL: http://www.enteract.com/~lspitz/audit.html. (14 June 2002).

[41] Spitzner, L. "Understanding the FW-1 State Table." 29 November 2000. URL: http://www.enteract.com/~lspitz/fwtable.html . (14 June 2002).

[42] Stevens, W. ibid, p. 226-7, 246-7.

[43] Postel, J.B. ed. (Information Sciences Institute, prepared for DARPA). "RFC: 793 Transmission Control Protocol" September 1981. (Retrieved from URL: http://www.faqs.org/rfcs/rfc793.html on 15 April 2002).

[44] Fyodor. "Nmap network security scanner man page." Date not available (but the man pages are always included in the nmap distribution. URL: http://www.insecure.org/nmap/data/nmap_manpage.html . (1 July 2002).

[45] Spitzner, L. "Understanding the FW-1 State Table." ibid.

[46] Brenton, C. & Wagner, R. "RE: Anyone else seeing TCP ACKs on port 80? - Not LION Worm." 26 April 2002. URL: http://www.incidents.org/archives/intrusions/msg08129.html. (25 June 2002).

[47] Security Administrator, Corning Incorporated - dnstech@corning.com. "RE: ACK packets from 199.197.130.21 & 199.197.135.21". Personal email communication. 25 June 2002.

[48] Stewart, J. "SANS Global Incident Analysis Center > 12/11/00 - 1200." 11 December 2000. URL: http://www.sans.org/y2k/121100-1200.htm. (25 June 2002).

[49] Benninghoff, J. & Romero, J. "SANS Global Incident Analysis Center > Report Date: March 14, 2001 -

1400." 14 March 2001. URL: http://www.sans.org/y2k/031401.htm. (25 June 2002).

[50] Baldwin, L. "SANS Global Incident Analysis Center > Report Date: April 3, 2001 - 1430." 03 April 2001. URL: http://www.sans.org/y2k/040301-1430.htm. (25 June 2002).

[51] Results gathered from URL: http://www.geektools.com and http://www.arin.net .

[52] Halpern, J., Convery, S, & Saville R. "SAFE VPN: IPSec Virtual Private Networks in Depth." Cisco Systems, Inc:San Jose, CA. 16 Aug 2001. URL: http://www.cisco.com/warp/public/cc/so/cuso/epso/sqfr/safev_wp.htm (1 August 2002).

[53] CERT Coordination Center. "CERT Incident Note IN-2002-04: Exploitation of Vulnerabilities in Microsoft SQL Server." 23 May 2002. URL: http://www.cert.org/incident_notes/IN-2002-04.html. (14 July 2002).

[54] Microsoft Corporation. "Microsoft Security Bulletin MS02-035: SQL Server Installation Process May Leave Passwords on System (Q263968). 10 July 2002. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS02-035.asp (22 July 2002).

[55] Microsoft Corporation. "Microsoft Security Bulletin MS02-034: Cumulative Patch for SQL Server". 10 July 2002. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS02-034.asp (22 July 2002).

[56] Internet Storm Center. "Port Reports for 1433" URL: http://isc.incidents.org/port_details.html?port=1433 . 01 August 2002). The results were randomly taken for a substantial period of time to demonstrate the massive number of scans to this port. The highest reports totals came on 20 Aug 2002 with 23% of the total records, to a relative low of 7% of the total records attributed to 1433 on 30 Aug 2002)

[57] Northcutt, S. et. al. "Chapter 11 - Design Fundamentals" Inside Network Perimeter Security: The Definitive Guide to Firewalls, VPNs, Routers, and Intrusion Detection Systems. New Riders:Indianapolis, IN. 2003. p. 317

[58] Snort stable rule set distribution, ibid.

[59] arachNIDS. "IDS252/DDOS_DDOS-SHAFT-SYNFLOOD-INCOMING" No date available. URL: http://www.whitehats.com/info/IDS252 (17 Sep 2002).

[60] Dietrich, S., Long, N., & Dittrich, D. "Analyzing Distributed Denial Of Service Tools: The Shaft Case" Proceedings of the 14th Systems Administration Conference (LISA 2000). USENIX: New Orleans, LA: December 3-8, 2000. URL: http://www.usenix.org/events/lisa2000/full_papers/dietrich/dietrich.pdf

[61] Brown, K. "RE: Strange "DDOS" traffic." 27 March 2002. URL: http://www.incidents.org/archives/intrusions/msg03695.html . (18 Sep 2002).

[62] Heyman, L. "Strange "DDOS traffic." 26 March 2002. URL: http://www.incidents.org/archives/intrusions/msg03681.html. (18 Sep 2002).

[63] Cisco Systems, Inc. "Strategies to Protect Against Distributed Denial of Service (DDoS) Attacks" URL: http://www.cisco.com/warp/public/707/newsflash.html . 17 February 2002. (23 July 2002).

[64] Internet Security Systesm. "PROTOS Remote SNMP Attack Tool." 4 March 2002.

http://bvlive01.iss.net/issEn/delivery/xforce/alertdetail.jsp?id=advise110 (July 2002).

[65] Dell, J. Anthony. "Adore Worm - Another Mutation" SANS Institute Reading Room. 6 April 2001. URL: http://rr.sans.org/threats/mutation.php . (9 September 2002).

[66] Beardsley, Tod. "Intrusion Detection and Analysis: Theory, Techniques, and Tools." 8 May 2002. URL: http://www.giac.org/practical/Tod_Beardsley_GCIA.doc (15 Oct 2002).

[67] SnortSnarf. Silcon Defense. Project URL: http://www.silicondefense.com/software/snortsnarf/ (12 November 2001).

[68] Newport, Brandon L. "GIAC Training & Certification: Level Two Intrusion Detection In Depth GCIA Practical Assignment, Version 2.7a" 8 May 2001. URL: http://www.giac.org/practical/Brandon_Newport_GCIA.zip. (3 June 2002).

[69] Wells, Garth. "Counting Transactions per Hour using a Pivot Table." SQL TEAM.COM. 9 September 2001. URL: http://www.sqlteam.com/item.asp?ItemID=5741 (31 July 2002).