The Null Server: A Model for Server Farm System Security

Robert J. Foertsch and Brian M. Slator

Computer Science and Operations Research

North Dakota State University

Fargo, ND 58108

rjfoertsch@gmail.com brian.slator@ndsu.edu

Abstract

The Null Server is a security model implemented for the World Wide Web Instructional Committee (WWWIC) at North Dakota State University (NDSU). An increased level of security is achieved by isolating important services necessary for the operation of the server farm onto one machine (the Null Server), and then using firewall rules to completely hide that machine from the Internet. The result is an inexpensive security arrangement that offers inherent protection against a range of potential attacks.

This paper tells the story of a PHP injection attack that led to new thinking on server farm security. The story concludes with a happy ending as a subsequent attack is defeated with somewhat comic results. This paper then discusses the Null Server model in detail and describes the various services and advantages that the Null Server provides.

1 Introduction

The Worldwide Web Instructional Committee (WWWIC) at North Dakota State University (NDSU) operates a small server farm for research purposes. Nine of these machines are running versions of a Linux operating system, one machine running Windows Server 2003, along with an eight-node Beowulf cluster (all running a Linux operating system). Each of these machines provide a mix of standard and proprietary services – one machine running both a Lightweight Directory Access Protocol (LDAP) server controlling user authentication and a network file server (NFS), and another server being used for basic remote system logging. Over the years the services on these machines have been attacked in a number of ways; resulting in one server being compromised in the spring of 2008. To defend against these incursions a system called the 'Null Server' has been implemented – a machine that provides services essential to operating a secure and stable server farm, yet is completely isolated from the outside world.

Originally, WWWIC employed a fairly typical server configuration with a key machine that served many centralized purposes. This machine not only hosted a web server with general information about WWWIC, it also acted as a full mail server for students and staff members, as well as acting as the LDAP authentication server and hosting the NFS for some home directories. Figure 1 shows a simple diagram of the original setup.

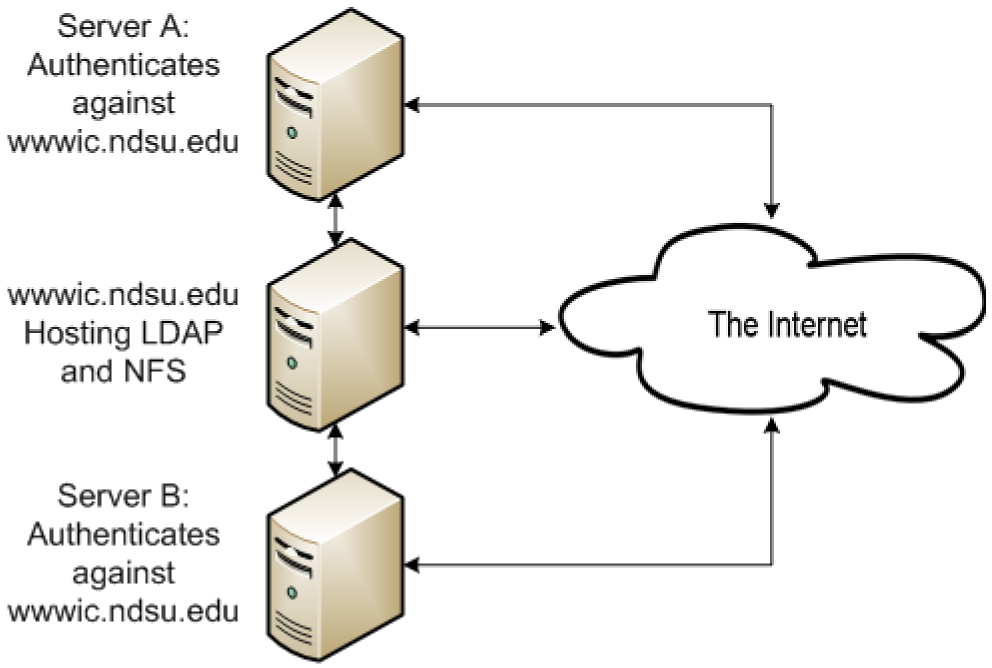

Server A:
Authenticates against wwwic.ndsu.edu
wwwic.ndsu.edu Hosting LDAP and NFS
Server B:
Authenticates against wwwic.ndsu.edu
The Internet

Figure 1: The original setup for the WWWIC server farm. Note that the server hosting the LDAP and NFS is directly connected to the Internet.

1.1 The Attack on Solidarność

In the month of February 2008, WWWIC received an email from the Information Technology Service (ITS) group on the NDSU campus. One of WWWIC's servers (specifically *javamoo.ndsu.edu*) was spamming the group Solidarność (as an aside, this is the Polish group Solidarity, the first non-commmunist trade union in any of the Warsaw Pact countries [1] – their leader, Lech Walesa, won the Nobel Peace Prize in 1983 and was the president of Poland, 1990-1995 [2]). Solidarność had contacted ITS, who then in turn blocked *javamoo.ndsu.edu* from the network. In order to maintain operations a temporary server was added to the network to serve the roles *javamoo.ndsu.edu* was fulfilling.

After an investigation, it was determined that through a vulnerability in PHP, the Apache server had executed a script that downloaded a modified version of EggDrop (a popular IRC bot [3]). Using the Apache user account, the script created a *cron* table entry for itself to keep it alive, and started spamming Solidarność (it would be safe to assume that javamoo.ndsu.edu was spamming many different machines; Solidarność was the first to contact NDSU to complain about the attack).

1.2 Problems and Solutions

One troublesome aspect of the 'Javamoo Incident' is that it was not a complex attack. Odds are, the remote machine that compromised *javamoo.ndsu.edu* is simply crawling servers on the network. It was not performed by an individual who wanted to explicitly break into that specific machine; it was simply a successful 'drive-by attack'. It was pure luck that *javamoo.ndsu.edu* was the machine that was compromised, and not *wwwic.ndsu.edu* (which was hosting the LDAP and NFS at the time).

Knowing this, WWWIC proposed the following hypothetical scenario: suppose a security hole in the Apache web server on the *wwwic.ndsu.edu* machine is exploited in a similar manner. What implications would that have for the rest of the server farm? To resolve the situation, the machine would need to be taken off the network so the administrators could investigate the problem and fix it; possibly requiring a reinstall of the operating system. However, because this key machine is also hosting the LDAP and NFS for the rest of the machines – none of the users can log in until the situation is resolved. Depending on the extent of the breach, the home directories on the NFS could be considered compromised, too. Worse yet, what if the security hole exploited allowed escalation to the root user (unlikely, but still certainly possible)? Then all of the LDAP account information could be considered compromised.

This worst case scenario is not far-fetched given the typical configuration found in so many institutions (as shown in Figure 1), and the ramifications of this scenario clearly show there are numerous potential problems associated with this common server setup.

To address this problem, WWWIC proposed a solution: the Null Server.

The Null Server exists to host many essential services required by the other machines, including LDAP for user authentication, NFS, Syslog system logging, and communication with an Uninterruptible Power Supply (UPS), along with various system monitoring tools and data backup utilities. The Null Server is also completely blocked from the rest of the Internet through firewall rules. It only allows machines on the local network to connect to it. Even OpenSSH (which is considered to be highly secure) requires that the connecting client must be on the specified subnet. This adds an important extra layer of security. Even if a vulnerability like the Apache bug still exists, it is extremely difficult to exploit because the machine simply cannot be accessed by the hackers on the Internet or their web crawlers.

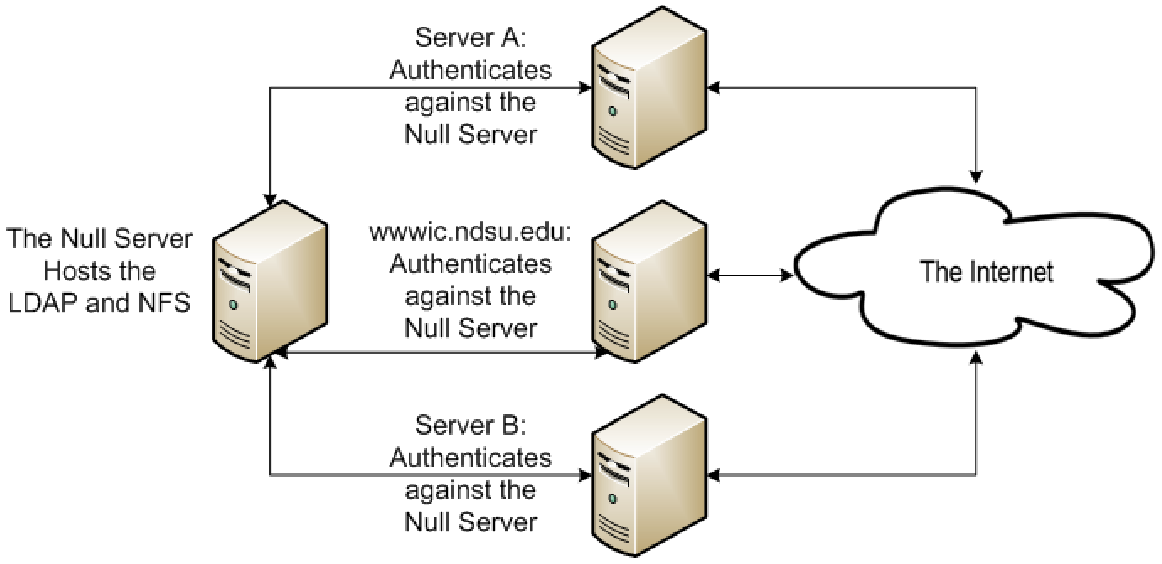

Server A: Authenticates against the Null Server
The Null Server Hosts the LDAP and NFS
wwwic.ndsu.edu: Authenticates against the Null Server
The Internet
Server B: Authenticates against the Null Server

Figure 2: The WWWIC server farm after implementing the Null Server system. Note that the server hosting the LDAP and NFS is NOT directly connected to the Internet.

Of course ensuring server security for an academic group such as WWWIC is a two-fold problem. We need a system that not only offers excellent security for the servers, but also can minimize cost.

The Null Server is an effective solution for both issues. It offers an excellent security solution simply because it can isolate services that are required by the various servers but do not need to be accessed from the Internet. The Null Server also does not require any expensive hardware (any older, 'retired' machine can be set up to provide the necessary services). All of the software solutions on the Null Server are based on free open-source software (Nagios, Tripwire, and the Lightweight Directory Access Protocol (LDAP) are either open source or have open source implementations).

2 Methods of Implementation

The key insight behind the Null Server model is that security is improved if centralized services like LDAP, NFS, and system logging are implemented on a machine visible to the other machines in the server farm but hidden from the Internet. This section describes the implementation and configuration of these centralized services.

2.1 Lightweight Directory Access Protocol

The Lightweight Directory Access Protocol (LDAP) as defined by the OpenLDAP project, is "... an open-standard protocol for accessing X.500 directory services. The protocol runs over Internet transport protocols, such as TCP [4]." LDAP is diverse, and can be used to model any data that can fit a 'Directory' paradigm. For example, the online phonebook at North Dakota State University uses LDAP to categorize attributes about a student, including major, year, email, etc. In the WWWIC server farm, LDAP is used to control account information and user authentication over multiple machines.

User information is stored in two places in a standard Linux environment. This example explicitly looks at the Debian based Linux distribution that is primarily used in the WWWIC server farm: Ubuntu. Linux distributions based on Red Hat Linux or other distributions may vary slightly. The */etc/passwd* file contains general information about a specific user (user name, user ID and group ID numbers, home directory and the default shell, etc.), and the */etc/shadow* file contains the hash of the user passwords. To properly authenticate users on a system, WWWIC's Linux systems use the Pluggable Authentication Modules (PAM). PAM modules can be used for many different purposes. The most common case is to authenticate a user (through a password) against the password hash stored in the */etc/shadow* file, and then permit or prevent login based on the results. However, PAM modules are flexible, too. If a certain group of users should only be able to log into a system during a specified time period (a typical example would be to allow login from certain users during normal working hours, weekdays from 9:00am until 5:00pm), there's a PAM module for that. PAM modules can also be used to ensure users have a strong password, or to prevent users from logging in when the */etc/nologin* file exists (typical when a server is preparing to shut down). Clearly, using PAM modules for user authentication is versatile.

PAM modules also exist allowing for an LDAP server to act as an authentication back end for user account information instead of the */etc/passwd* and */etc/shadow* files. In this configuration, the LDAP server maintains all of the user data (including password hashes) and the client machines authenticate against the LDAP server to see if a user should be given access to a machine or not. This offers many advantages when compared to the standard Unix system of user authentication, especially when a server farm grows in size:

1. User account information is controlled from one central point. For a large server farm, the standard Unix system for user accounts becomes difficult to work with. Even 100 accounts on 10 machines would result in 1000 different user accounts. If an administrator wishes to add a user to a server, it is as simple as adding a user name to an LDAP group. Without LDAP, an administrator must log into each specific machine, add that user, and create a new password for that account. If the user was given access to ten machines in the WWWIC server farm, the administrator would need to duplicate the effort on every single machine; such a process can be difficult and error prone.

2. Through an LDAP authentication system, user information is no longer stored on the client machines. A common trick of rogue hackers is to get the list of users from the */etc/passwd* file (which is readable by all users); this greatly increases the success rate of a brute force attack, since the rogue hacker has valid accounts to test. If an exploit give access to the root account, the hacker would then be able to access the list of hashes from the */etc/shadow* file (readable only by the root user). Depending on the hashing or encryption method used, the rogue hacker could reverse the hashes using a Rainbow Table [5]. Using LDAP, if a machine is compromised, the hacker can no longer gain information from the */etc/shadow* or */etc/passwd* files.

2.1.1 LDAP and the Null Server

The use of an LDAP server is enhanced with the Null Server system. Originally (before the spring of 2008), the LDAP server was running on *wwwic.ndsu.edu*, which was a server accessible from anywhere on the Internet. This is problematic for two reasons.

First, anyone could access the information stored in the LDAP database (excluding password information). This is the default configuration in LDAP, but this allows anyone to get a list of user names along with the specific machines they have access to. So an individual could increase the success rate of a brute force password attack since they would be aware of valid user names (many brute force attacks test against common user account names). An administrator can minimize this risk by enforcing users to have a strong password. Side note: North Dakota State University currently uses LDAP as the mechanism for users to log into a computer cluster, check their email, etc. An individual could quickly gain access to a full list of students and user names if they know the domain name or IP address of these LDAP servers.

One way to reduce the risk imposed by the previous problem would be to simply use various firewall rules to allow specific clients to connect. LDAP uses port 389, and LDAPS (LDAP with TLS encryption) uses port 636. An administrator could easily restrict access to those two ports from a specific list of IP addresses, or a subnet. This would allow the local machines to connect to the LDAP server and authenticate a user, but still prevent the list of users from being accessible from anywhere on the Internet.

Unfortunately, using a firewall to limit authorized clients to connect and authenticate users does not solve the second problem. The second reason why WWWIC's original setup was ineffective is because multiple services exist on *wwwic.ndsu.edu*, and compromising any of those services could cause an unscheduled downtime or instability for the server. As stated earlier, *wwwic.ndsu.edu* was running multiple services, including an Apache web server and Dovecot for IMAP and POP3 mail services. If a vulnerability in any service was exploited, the machine would need to be offline while the situation is resolved. Since *wwwic.ndsu.edu* would also be serving as the LDAP authentication server, users would not be able to log into any of the other systems until a replacement LDAP server was brought online. If an exploit was severe enough (allowing escalation to administrative privileges), the entire LDAP database could be considered compromised, including the user names and password information – clearly an unfortunate situation.

As mentioned before, the Null Server is a preventative measure against such a situation. The first reason listed above is quickly fixed with a firewall rule. The Null Server does not allow any machines outside the local network to connect to it through any port. So an individual cannot gain access to the list of user accounts because that individual simply cannot access the server at all.

The second problem listed above is also effectively alleviated. The sole purpose of the Null Server is to isolate services that – while essential – do not require full access to the Internet. These services, including LDAP, are only needed by machines within the lab on the local area network. Therefore they can be isolated on the Null Server, greatly reducing the chances that an exploit is successful while still providing a high level of security and stability for the server farm.

2.1.2 LDAP and User Administration

LDAP also simplifies user administration. LDAP uses a tree directory structure to order information. Groups are called 'Organizational Units' (*ou*), and within each group are entries called 'Common Name' (*cn*).

WWWIC have three primary organizational units: Devices, Group, and People. The 'Devices' group corresponds to one machine (or a group of machines). 'Group' corresponds to the typical Unix group structure, and 'People' corresponds to individual users. Figure 3 shows a diagram of the current setup.

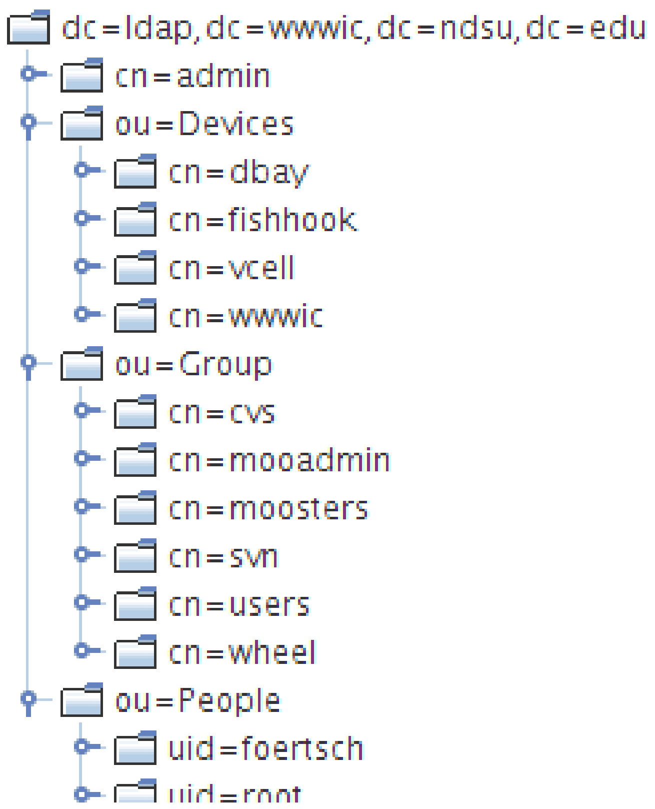


Figure 3: A simplified representation of the WWWIC LDAP tree structure hierarchy

If a new student volunteers to help with WWWIC's Virtual Cell project, for example, an administrator simply needs to create a new entry under the People *ou*, and add him or her to the list of accepted users for the *vcell* Device. If another student who has worked on many of WWWIC's projects (therefore has been granted access to many of the machines) graduates from college, an administrator can disable that student's account under the People *ou* (instead of logging in and disabling the account on every machine that the student had access to).

2.2 Linux Logging Services

Effective system log management is one of the most important aspects to consider when developing a stable and secure server farm. Unfortunately, it is also one of the aspects that is often overlooked in a server environment. This section discusses the importance of log file management, and looks at the simple implementation currently in use with WWWIC's Null Server configuration.

2.2.1 The Importance of Logging

Logging is one of the services on a Unix system that is often overlooked. There are two primary reasons for this.

First, logging is never the primary reason for the existence of a server. When a machine is brought online to host a web server, it should be clear that the purpose of the machine is to host web pages. If the machine can host web pages correctly, why should a system administrator worry about the system logging facilities?

The second reason logging is often overlooked is that logging is often a passive solution to a problem. More specifically, system logging is perfect at identifying why a certain problem is occurring; conversely, if errors are not occurring, then a log file is far less useful. The classic example is Apache's *error.log* file (by default located here: */var/log/apache2/error.log*). If a CGI script is not executing correctly, the *error.log* file is instrumental in identifying and correcting the problem. If a CGI script is executing flawlessly and has never been problematic, then the *error.log* file is not useful (assuming an error will never occur in the future).

A perfect example of the true value of log files can be found in an incident that occurred in a student Linux lab offered by the NDSU Computer Science Department. Over the winter break in 2009, an unauthorized individual entered the lab. This individual then booted one of the machines in single-user mode, tried to log into a remote machine used for user authentication, but failed (the user authentication server was located in a different room; the lab modeled the LDAP system after WWWIC). The individual then wiped specific log entries from the lab machine, and left.

There are two things to note from this scenario. First, the individual that broke into the lab was skilled. The intruder didn't simply wipe all of the log files (such an action would be very obvious; missing parts of a log file is more difficult to discover than completely removed log files). It appears that Unix commands such as *grep* and *sed* were used to remove specific entries about the break-in, but not other log entries. The second item to note is that the only reason this break-in was found was because of the irregular login attempts found in the LDAP server log files.

In any case, it is highly desirable that a thorough, well defined log file policy must be established and implemented for any server environment. However, the default configuration is often not sufficient for adequate logging needs. For example, WWWIC primarily uses Ubuntu 8.04 Server Edition as the operating system of choice for the Linux servers. The default configuration will delete log files after two months (Ubuntu uses *logrotate* to rotate the log files). This is simply not enough 'history' for a production server. Especially with the decreasing costs of storage space and the efficient use of compression of the text logs, there is no reason not to keep logs archived indefinitely. (This statement ignores the fact that many institutions or groups may have implemented a privacy policy towards keeping log files. This is especially true in the case of a web server, which can log the IP address of any machine that connects to it. However, it is possible to anonymize log files after a certain date.)

2.2.2 Remote Logging

In addition to maintaining log files locally, Syslog (which is one of the utilities to log data on a Unix system) can be configured to remotely receive log data. Remotely receiving logs should be an essential part to a server environment, and a role that the Null Server fulfills seamlessly.

Why is remote logging essential? If a user is able to gain administrative access to a server, it is a very simple act to delete log files to cover their tracks. An administrator should be aware that a system has been tampered with (missing log files would be the first clue), but it would be very difficult to determine what actually happened. The primary diagnostic tool an administrator would use is the log files, but the log files have been deleted.

If a server is configured to log data both locally and remotely, this problem is quickly solved. Syslog has the functionality to send log files over the network (Syslog can also encrypt the information sent over the network in the event the network is not trustworthy [6]). Every time an event is logged to one of the files, the same data is transmitted over the network to a logging server configured to receive log data.

Therefore, the problem is solved. If a rogue hacker gains administrative access to a machine and the data is logged, even if that user deletes the log files of the local machine, that information is duplicated on the Logging server. In the case described above when an individual broke into the student Linux lab, remote logs would have proven invaluable to show what activities truly happened. The malicious activity would be quickly identified and resolved by looking through the files located on the remote logging server.

It should be obvious that the Null Server is an excellent candidate to serve as a Logging server. It would be unwise to give a logging server full access to the Internet, simply because such an essential service should be completely defended from attack. In fact, a logging server should be the most secure machine of all because it contains all of the diagnostic information of the server farm. Again, since the Null Server is completely cut off from the Internet, it can be configured to act as a Logging server, too.

2.2.3 Log Monitoring

In addition to simply archiving log files, it is advantageous to view and analyze log files for suspicious activity. Unfortunately, directly reading all information sent to the system log files is a very tedious and time consuming process; it is also difficult to identify patterns in such a large set of data. This is where a log monitoring or a log reporting tool can be very useful.

WWWIC uses Logwatch to provide summaries of log files. Logwatch "is a customizable log analysis system. Logwatch parses through your system's logs for a given period of time and creates a report analyzing areas that you specify, in as much detail as you require [7]." It was created by Kirk Bauer to provide a simple way to automate the analysis of log files.

A typical Logwatch report contains many important pieces of information: a count of successful and failed login attempts, suspicious Apache HTTP requests, general mail message statistics including total messages sent along with the aggregate message size, SMART (Self-Monitoring and Reporting Technology System [8]) drive statistics, overview of commands executed with the *sudo* command, disk space remaining, and system uptime. Logwatch can be configured to report on additional log characteristics, too.

Here is a simple example to show the value of using Logwatch. On March 9th, 2009, the machines in the WWWIC lab were subjected to a very thorough network scan, identifying services and trying to exploit potential vulnerabilities on each machine (for example, according to Logwatch, over 2000 different known URL attacks were used against the Apache server running on wwwic.ndsu.edu). The scan originated from one specific machine with a NDSU IP address. The system administrator for WWWIC's machines was alerted within 24 hours of the scan, and was able to alert security officials about the scan and could identify the machine from which the scans originated.

As it turns out, it was NDSU's IT security personnel who originated the scan, looking to identify insecure and out of date machines on the NDSU network through the use of an unannounced attack. Another interesting fact is that since the Null Server is completely isolated by firewall rules, ITS was unable to determine any services that exist on it. Without Logwatch, it is doubtful that the scan would have been identified at all. Thus, the Null Server even protects you from attacks by your own security officials.

2.3 The Network File Server

A network file server (NFS) is a very simple idea. On a standard Unix system, an administrator can *mount* a drive partition or directory to a specific file system location. It is common to see the root directory (*/*) as one drive partition, and the home directory (*/home*) to be mounted as another drive partition (or a different drive altogether).

A NFS takes that same idea, but instead of mounting a partition of a drive connected to the machine, an administrator can mount a partition or directory from a remote machine running the NFS daemon.

NFS is a service that obviously needs access to the machines looking to mount its directories. But since it doesn't need access to the rest of the Internet, NFS fits the model for another service to add to the Null Server.

Network file servers are simple, and do not need a lot of discussion compared to LDAP or remote logging. Nonetheless, it is a good idea to be restrictive on what machines are allowed to mount a remote directory. Restrict such privileges to the specific machines that are authorized, not a group of machines or a subnet. If restricted to a subnet, an individual could mount the directories without authorization, as long as they can plug into the subnet. Since it is assumed that the individual has full administrative access to the machine that was plugged into the local network, he can view all of the files on the NFS, regardless of file permissions. NFS allows an administrator to only allow access to certain machines through the */etc/exports* file on the server.

3 Additional Services

In addition to LDAP for user authentication, the remote logging functionality and the NFS, the Null Server model has added a few additional opportunities for services that would not have been applicable in the original WWWIC model. This section goes over these services in detail.

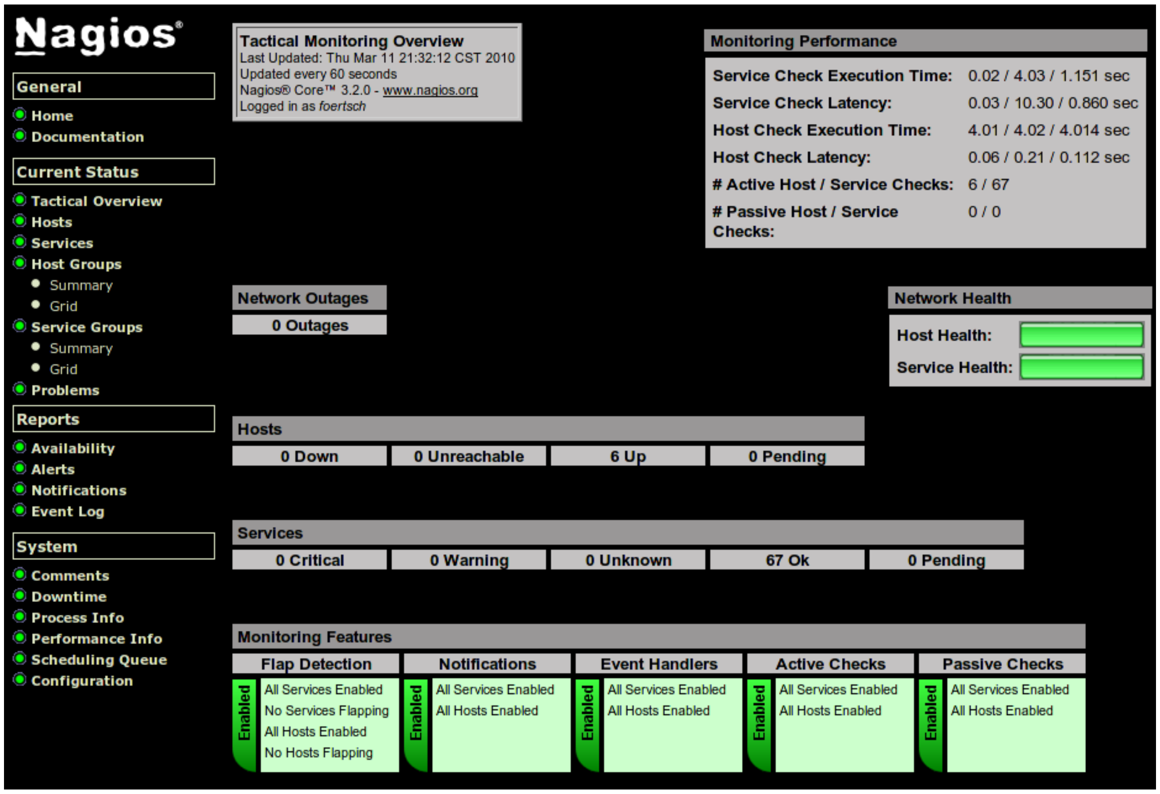
From this point onward, this paper discusses a variety of recommendations for server farm management, independent of the Null Server model and common to any server farm. These services have value added to them, however, though the use of the Null Server.

3.1 Nagios

Nagios is an award winning open-source system monitoring application [9], and is recommended for server farms of any size, local or distributed. WWWIC began using Nagios to monitor its servers in the spring of 2009.

Nagios is another very simple idea. The Nagios daemon (through the use of a plugin) can check the state of a machine and records the resulting information in a data file. Through the use of multiple plugins, Nagios can monitor any metric of a system (for example: system load average, number of users currently logged in, memory usage, whether updates are available through *apt-get* or *yum*, etc.), and an administrator is able to create additional plugins to suit the specific needs of a server farm. Nagios can also determine information on remote machines, too (the Nagios server is the one running the daemon that is getting updated statistics, and the clients are machines where the statistics are being derived from) through the use of OpenSSH keys or Nagios Remote Plugin Execution (NRPE). Plugins are versatile, and easy to write. For example, WWWIC now has custom made Nagios plugins that can:

1. Check the state of the various multi-user dungeons (MUDs) running on WWWIC servers [10].
2. Check the state of the Folding@Home process running on a few of WWWIC's machines. In addition to seeing whether the Folding@Home process is running, it can also state the percentage complete on the current work unit.
3. Check the temperature reading from the machine's hard drive(s).



Tactical Monitoring Overview
Last Updated: Thu Mar 11 21:32:12 CST 2010
Updated every 60 seconds
Nagios® Core™ 3.2.0 - www.nagios.org
Logged in as *foertsch*

Monitoring Performance

Service Check Execution Time:	0.02 / 4.03 / 1.151 sec
Service Check Latency:	0.03 / 10.30 / 0.860 sec
Host Check Execution Time:	4.01 / 4.02 / 4.014 sec
Host Check Latency:	0.06 / 0.21 / 0.112 sec
# Active Host / Service Checks:	6 / 67
# Passive Host / Service Checks:	0 / 0

Network Outages
0 Outages

Network Health
Host Health:
Service Health:

Hosts

0 Down	0 Unreachable	6 Up	0 Pending

Services

0 Critical	0 Warning	0 Unknown	67 Ok	0 Pending

Monitoring Features

Flap Detection	Notifications	Event Handlers	Active Checks	Passive Checks
Enabled: All Services Enabled, No Services Flapping, All Hosts Enabled, No Hosts Flapping	Enabled: All Services Enabled, All Hosts Enabled	Enabled: All Services Enabled, All Hosts Enabled	Enabled: All Services Enabled, All Hosts Enabled	Enabled: All Services Enabled, All Hosts Enabled

Figure 4: The Nagios Tactical Overview page. All hosts and services are performing correctly.

The extensibility of Nagios makes it a powerful tool to have in a server farm. If a service or a state can be determined by a script, then it can be monitored by Nagios. As of the writing of this paper, the Nagios system on the Null Server checks 338 service states and statistics on 10 different machines (9 Linux servers and one Windows server). Nagios can check all 338 services in under 10 minutes. WWWIC also operates a small Beowulf cluster; Nagios checks 67 services on the eight nodes.

In addition to simply knowing the state of a service, Nagios can also perform actions based on the result of a plugin. The typical example is an email or an SMS alert. If a machine were to became unresponsive, or if a particular service went offline, an administrator would receive an alert within 10 minutes of that event occurring. In addition to an email or SMS alert, Nagios can also execute a command based on the output of a plugin. Here is a classic example where such a feature is invaluable: Suppose a Computer Science department runs a Tomcat server for student projects. A student would be able to bring down the Tomcat daemon with some poorly designed code. Instead of waiting for an administrator to recognize that Tomcat is no longer running, Nagios can be configured so that when Tomcat plugin returns a critical alert, Nagios simply restarts the Tomcat daemon. (One may identify a possible security problem here, but it can be avoided through good system administration practices. Nagios never runs as the administrative user; restarting a service such as Tomcat would require root privileges. Any security concerns can be alleviated through the proper use of the *sudo* command.)

Nagios offers many advantages in a server farm environment:

1. Nearly instantaneous alerts when problems arise. In server farms of any size, it may be easy to forget about a service running on one machine. Without any sort of monitoring system in place, it may be two or three days before anyone notices that it is not running, and even then, an administrator would not know the time or date that it stopped running without digging deep into the log files. With Nagios, an administrator will know within 10 minutes if a service is operating outside acceptable parameters or if a machine is down (sooner, depending on the specific configuration).

2. Nagios can also help identify specific times of high system load. Many of the MUDs running on WWWIC servers are used for educational purposes by local school districts. By using Nagios, an administrator can look back through previous events to see how high the load average (or other important metrics) get during these times of high stress. An additional application can be installed with Nagios that provides graphs of system metrics, too.

3. When a machine is compromised, or when hardware begins to fail, it may not be immediately apparent. Through a system monitoring application like Nagios, you can quickly detect any and all anomalous behavior in the system and address the issues.

The Null Server offers a perfect environment to host Nagios. As mentioned earlier, Nagios can execute commands on the client machine either using password-less OpenSSH keys or through NPRE. Every system administrator should cringe when discussing remotely executing commands on a server; such a system has many security implications that need to be addressed to properly secure a system. Using the Null Server to host Nagios helps alleviate those problems – having world accessible machines logging into each other through password-less OpenSSH keys can be reckless. If one machine is compromised, a skilled hacker could simply gain access to the other machines using the password-less key. The Null Server is assumed to be more secure than the other machines, greatly reducing the risk.

3.2 Tripwire

Tripwire is an open-source Intrusion Detection System (IDS) [11]. The main purpose of an IDS is to monitor and identify modified files on a system. Files that are essential for the stable and secure operation of a system should never be tampered with; even if a rogue hacker gains administrative access to a system, any modified files are quickly identified and reported. The most important aspect of an IDS is that it does more than identify when files have been modified, the opposite is also true. If no warnings occur when scanning files, the administrator is guaranteed that those files have not been modified. Without an IDS, it is impossible to guarantee that a file has not been modified; root kits are designed to be undetectable through regular means (a root kit is application intended to hide the fact a system may be compromised [12]).

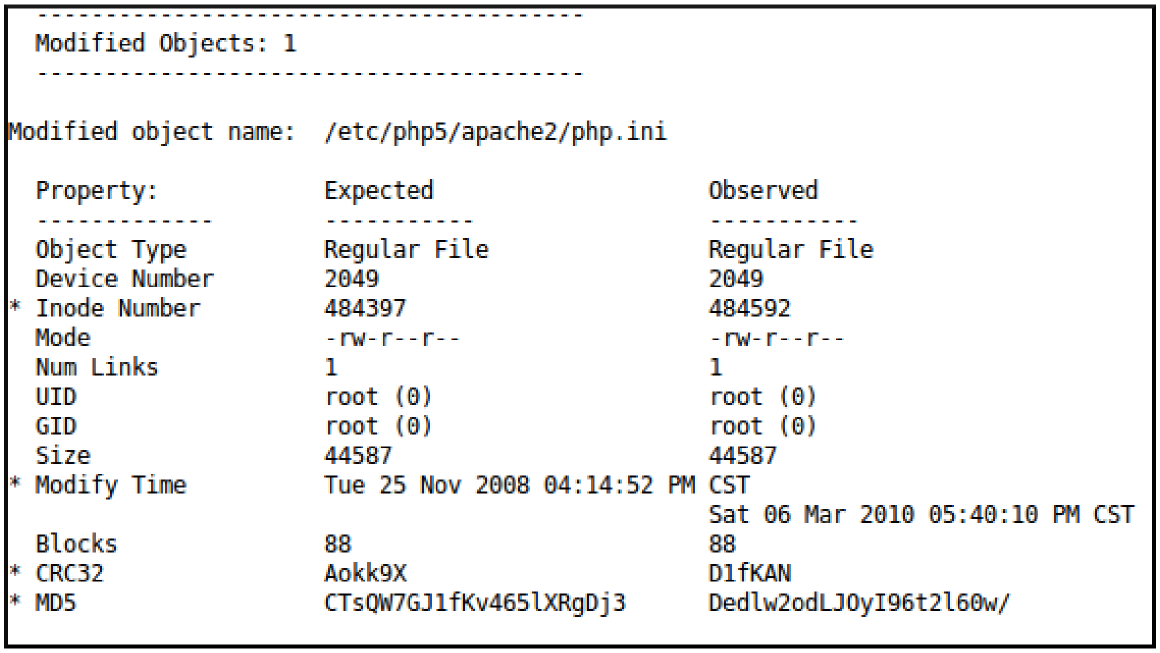
Intrusion Detection Systems are a must have for server environments, and are simple to implement once an administrator understands how they work.

Tripwire maintains a database file of characteristics about files (it can be any file, but generally it is aimed more toward system essential files that are targets for root kits). These characteristics include file hash results, access and modification dates, size, permissions, etc.

Tripwire can monitor a wide variety of file attributes. How can this be useful when considering system security? Imagine a situation where a rogue hacker gains administrative access to the machine, and leaves behind a root kit. In this example, it is assumed the root kit modifies the *ps* command. When an administrator executes *ps* to show the current list of processes, any malicious processes are omitted from the listing in the modified command. Now an administrator cannot detect the malicious process running because the very tools used to detect it have been tampered without his knowledge.

This is precisely the situation where Tripwire is useful. In a normal situation, the *ps* command file should never change (this scenario is obviously ignoring a system upgrade, but we make the assumption that a good systems administrator knows when he/she is upgrading a system). More precisely, attributes such as the file modification timestamp, permissions, file size, and the various hash results from the content of the */bin/ps* file should never change. If the system has been tampered with and the *ps* command has been modified, Tripwire will detect it since the values of the file are now different from the accepted values stored in Tripwire's database file. In addition to alerting the administrator that the file is different, Tripwire will also inform the administrator about which attributes have been changed. The administrator can then take the necessary steps to investigate and re-secure the system.

Tripwire is also versatile enough to allow for different attributes to be monitored for different files. By modifying Tripwire's policy file, an administrator can set different attributes to be watched for different files and directories. On a Unix file system, the



```
  -------------------------------------
  Modified Objects: 1
  -------------------------------------

Modified object name:  /etc/php5/apache2/php.ini

  Property:            Expected                  Observed
  -------------        -----------               -----------
  Object Type          Regular File              Regular File
  Device Number        2049                      2049
* Inode Number         484397                    484592
  Mode                 -rw-r--r--                -rw-r--r--
  Num Links            1                         1
  UID                  root (0)                  root (0)
  GID                  root (0)                  root (0)
  Size                 44587                     44587
* Modify Time          Tue 25 Nov 2008 04:14:52 PM CST
                                                 Sat 06 Mar 2010 05:40:10 PM CST
  Blocks               88                        88
* CRC32                Aokk9X                    D1fKAN
* MD5                  CTsQW7GJ1fKv465lXRgDj3    Dedlw2odLJOyI96t2l60w/
```

Figure 5: Typical output from a Tripwire report; this shows that a PHP configuration file has been modified.

/bin, */sbin*, */usr/bin*, and */usr/sbin* directories should never change without an administrator's knowledge. Files located in */var/log* will have different hash values and file sizes between two Tripwire executions, but the permissions should never change. Tripwire can be fine tuned to any system configuration.

Tripwire is also known for having an excellent commitment to security. Tripwire's policy file, configuration file, and the database and report files can all encrypted by a password (it is strongly recommended that the encryption password is not the administrative password for obvious reasons), so those essential files cannot be modified without knowing the password. But even with these default security provisions, WWWIC's use of the Null Server allows it to be taken one step further.

In the WWWIC server farm, the Tripwire configuration, policy, and database files all exist on the Null Server, not on the individual machines. On a daily basis, the Null Server will transfer the proper files to the machines in the server farm, execute a Tripwire check (comparing individual files on the server against stored file attributes in the Tripwire database file), and copy the results back to the Null Server, removing any traces from the individual machines. This setup offers two key advantages.

1. Theoretically, a skilled intruder might be able to access and modify the Tripwire binary files directly. A root kit could then in theory modify the Tripwire binary so that modified files do not show up in the Tripwire report. By having the Tripwire files on the Null Server (which is assumed to be highly secure), such a risk is minimized.

2. If a user gains access to one of WWWIC's other systems, it would appear that Tripwire is not used on the system (since the Tripwire configuration, policy, and binary files are stored on the Null Server, not on the individual machines). The user would then think that he/she can modify files without being detected, which is a trap we have set for them.

Therefore, through combining the intrusion detection abilities of Tripwire, along with the secure environment that the Null Server provides, WWWIC's machines are fortified against incursions with root kits.

3.3 UPS Monitor

In the WWWIC server farm, the Null Server is able to provide additional system stability through the use of the APC UPS daemon.

WWWIC currently has an Uninterruptable Power Supply (UPS) (specifically, the APC Back-UPS Pro 1400 model UPS device) to supply battery power to the machines when the building's power is cut. UPS devices are very useful to have in a server environment, by serving two different purposes. First of all, a UPS device is able to provide a more consistent stream of electricity to a server. Just like a surge protector can protect a machine from fluctuations in the current, a UPS device can provide that same functionality. Secondly, a UPS can protect a machine against sudden power outages. A sudden power outage or a brownout can cause many problems for a server, including damage to the hardware along with the risk of data loss from an ungraceful shutdown.

Initially, when the WWWIC server farm was created, the servers were connected to the UPS device, and that was it; no additional configuration was completed. So if a power outage occurred, an individual would have to manually shut down each machine. If someone was not in the lab to perform this task, the UPS battery would eventually be depleted, and the machines would suffer the same immediate power loss that would occur if the they were not connected to the UPS. This would have the same risk of hardware damage and data loss; clearly not an optimal solution,

Luckily for WWWIC, the UPS that was purchased possessed a serial communications port on the back of the device. Since the UPS could communicate with only one machine, WWWIC's Null Server was an obvious choice. To communicate with a UPS device and to execute events when certain conditions are met, the Null Server runs a daemon known as *apcupsd*.

Apcupsd has several different states that can be monitored. Each state is paired with an additional script that can be executed when that state is entered. Table 1 shows a list of the possible UPS states.

State	Description
changeme	The UPS device detects that the battery has reached the end of its lifespan and should be replaced.
commfailure	Communication with the UPS device fails.
commok	Communication with the UPS device is restored.
offbattery	Power is restored to the UPS.
onbattery	A power failure has occurred and the UPS is currently providing battery power to the servers.

Table 1: A list of the five APC UPS Monitor states.

The two states that are of obvious importance are the *onbattery* and *offbattery* state. When the UPS device enters an *onbattery* state, a script is executed, instructing all of the machines in the server farm to shut down in two minutes (with the exception of the Null Server, which is instructed to shut down in four minutes to guarantee the other servers are now off). If the *offbattery* state is not entered within two minutes, the machines will begin to shut down gracefully. If the *offbattery* state is entered within two minutes, the shut down sequence is canceled. An additional note: the BIOS of each machine in the server farm is configured to turn on when power is restored.

The set up described above is clearly advantageous. In the original server set up, an individual would need to be in the lab to manage a power failure. That individual would need to manually turn off the machines before the UPS battery is depleted, and would need to manually turn the machines back on once power was restored. In this new configuration, there is absolutely no intervention required when a power failure occurs. The UPS device interfaces directly with the Null Server, which is able to shut down the machines. Once power is restored, the machines are turned back on, again, without intervention by an administrator – thus increasing the system stability.

4 Discussion

The main point to realize is that every service running on every machine can be divided into two different groups: public services and private services. Public services are ones that can only function properly by having full access to the Internet. Mail and web servers are the classic examples. A mail server is of little value if individuals cannot access the machine to either send or receive mail. Private services are ones that only need access to a small subset of machines to function correctly. Services such as LDAP, remote logging, NFS, database servers, etc. are ones that fit into this category. Only a certain number of machines should ever need to contact the WWWIC LDAP server, so it is wise to create a server environment that enforces that specific restriction.

Of course, the Null Server does not need to be one machine, either. Depending on the size of a server farm, it might be wise to split up some of the private services onto different machines. Especially in an environment where older, retired servers are fulfilling these roles, it might be wise to place the LDAP authentication server onto one machine, and the remote logging server on a different machine. A Network File Server could be configured with multiple hard drives for increased amounts of storage and RAID support, and a database server could be optimized for serving requests. When institutions have more than enough old machines; the Null Server model allows those machines to serve an important role.

5 Conclusion

The Null Server is a simple, yet exemplary security model. It can offer a stable, secure server environment while still saving costs and time in the long run.

Security is enhanced through isolating specific services. The services that should execute on the Null Server are ones that are required by machines on a specific network, but do not need access to the entire Internet. The Null Server should never host services that need to be accessed over the Internet. By isolating those specific services, an administrator can then use firewall rules to restrict access to that machine from only authorized servers, and completely block the rest of the network.

The Null Server model is also cost effective. An LDAP server does not have high-end hardware or software requirements. Many older machines can easily serve that role. Services like Nagios, Tripwire, and remote logging can save time in the long run by identifying problems quickly (possibly even before they arise). They can also provide statistics about the state of a server farm, providing detailed information and trends about network activity and memory usage. This information can be used to help optimize machines for the roles that they provided.

6 Acknowledgements

The authors wish to thank Otto Borchert, Guy Hokanson, and Brad Vender for their helpful comments on this paper. We also thank Adam Helsene for providing details about the student Linux lab incident.

This research was sponsored in part by the National Science Foundation grant #GEO-0608082 and ND-EPSCoR.

7 References

[1]Solidarity (Polish Trade Union), Wikipedia [Online]. Available: http://en.wikipedia.org/wiki/Solidarity_(Polish_trade_union) [Accessed: Mar. 15, 2010].

[2]Lech Walesa – Biography, Nobelprize.org [Online]. Available: http://nobelprize.org/nobel_prizes/peace/laureates/1983/walesa-bio.html [Accessed: Mar. 17, 2010].

[3]Eggheads.org – Main Page, Eggheads.org [Online]. Available: http://www.eggheads.org/ [Accessed: Mar. 17, 2010].

[4]What is LDAP, OpenLDAP [Online]. Available: http://www.openldap.org/faq/data/cache/29.html [Accessed: Mar. 1, 2010].

[5]Atwood, Jeff. Rainbow Hash Cracking, Coding Horror, Sep 7, 2008. [Online]. Available: http://www.codinghorror.com/blog/2007/09/rainbow-hash-cracking.html [Accessed: Mar. 10, 2010].

[6]Gerhards, Rainer. Encrypting Syslog Traffic with TLS (SSL), May 6, 2008. [Online]. Available: http://www.rsyslog.com/doc-rsyslog_tls.html [Accessed: Mar. 10, 2010].

[7]Bauer, Kirk. Logwatch.org – Main Page [Online]. Available: http://www.logwatch.org/index.html [Accessed: Mar. 10, 2010].

[8]Smartmontools – Main Page, Sourceforge.net [Online]. Available: http://sourceforge.net/apps/trac/smartmontools/wiki [Accessed: Mar. 17, 2010].

[9]Network Monitoring Application of the Year, LinuxQuestions.org [Online]. Available: http://www.linuxquestions.org/questions/2009-linuxquestions.org-members-choice-awards-91/network-monitoring-application-of-the-year-780663/ [Accessed: Mar. 10, 2010].

[10]Borchert, Otto, Lisa Brandt, Guy Hokanson, Brian M. Slator, Bradley Vender, Eric J. Gutierrez (2010), Principles and Signatures in Serious Games for Science Education, in Gaming and Cognition: Theories and Practice from the Learning Sciences Edited by: Richard Van Eck. IGI Global. pp. 315-341.

[11]Open Source Tripwire, Sourceforge.net [Online]. Available: http://sourceforge.net/projects/tripwire/ [Accessed: Mar. 10, 2010].

[12]Rootkit, Wikipedia [Online]. Available: http://en.wikipedia.org/wiki/Rootkit [Accessed: Mar. 17, 2010].