

Self Port Scanning Tool: Providing a More Secure Computing Environment Through the Use of Proactive Port Scanning

Joshua E. Kocher
UC Berkeley
jkocher@berkeley.edu

Dr. David P. Gilliam
Network and Computer Security Group, Jet Propulsion Laboratory
dpg@jpl.nasa.gov

Abstract

Secure computing is a necessity in the hostile environment that the internet has become. Protection from nefarious individuals and organizations requires a solution that is more a methodology than a one time fix. One aspect of this methodology is having the knowledge of which network ports a computer has open to the world. These network ports are essentially the doorways from the internet into the computer. An assessment method which uses the nmap software to scan ports has been developed to aid System Administrators (SAs) with analysis of open ports on their system(s). Additionally, baselines for several operating systems have been developed so that SAs can compare their open ports to a baseline for a given operating system. Further, the tool is deployed on a website where SAs and Users can request a port scan of their computer. The results are then emailed to the requestor. This tool aids Users, SAs, and security professionals by providing an overall picture of what services are running, what ports are open, potential trojan programs or backdoors, and what ports can be closed.

1. Introduction

The Jet Propulsion Laboratory (JPL) desired to have available a tool for System Administrators (SAs) to scan for open ports on a computer system and compare the scan results with a description of standard services that use those ports by operating system. The intelligence gathered by the tool matched what ports were considered safe to have open for services that used them and what open ports could pose a risk. Many times ports are open on a computer by default, while the user or SA is unaware

or does not have knowledge of what service(s) may be using them. A tool was developed to address this problem to aid the SA in assessing what ports are needed for essential services, and what services can be safely turned off and what ports can be closed. The tool that was developed is an enterprise, port scanning tool accessible through a web interface. Reports generated from the port scan are then emailed to the SA and the User of that computer system to ensure that only authorized people are provided the information.

Most of the intrusions into computer systems, whether by a worm/virus/trojan program or an individual, are accomplished by exploiting a service or application vulnerability on a system through an open port. Often, the user is unaware of what services are running. Some of these services may not be needed and can be shut down. To do this, one needs knowledge of what services are running over which ports and what the services are, and which services can be shut down or disabled.

One example is the default installation of Windows, which shares the hard drives across the Server Message Block (SMB) protocol. Though these shares are unable to be seen from another windows computer, they can be accessed directly with Windows or listed with the Linux implementation of the protocol, SAMBA [1]. Although a password is required to access administrative shares on a Windows system, passwords can be compromised, thus making the open network share a security risk. If the administrative share is not needed, the administrative shares can be disabled. By scanning systems for open ports and identifying the services using those ports, SAs and users can be made aware of potential security risks and take steps to mitigate them. The

port scanning tool developed at JPL improves provides for this capability.

2. Overview of the Self-Port Scanning (SPS) Tool

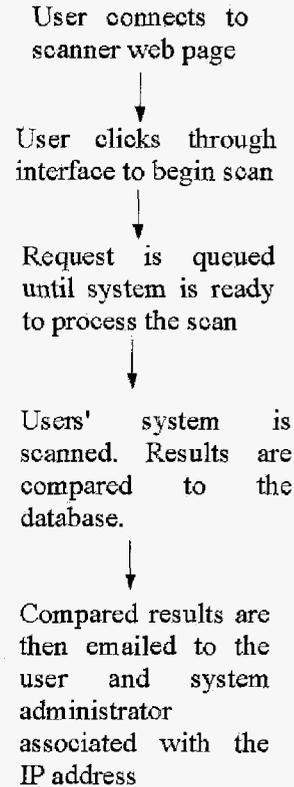
The Self Port Scanning (SPS) tool is basically an interface to nmap [2], a de-facto standard of port scanning tools. The user of the tool has the option to perform a port scan and have only the results of the scan emailed to them or they can have the results emailed to them along with baselines, allowing them to compare their scan results to the baselines. The baselines are a list of ports and services using those ports with a severity rating of running a particular service on a given operating system.

When a system is requested to be scanned the user is required to authenticate through the web server hosting the SPS tool. The SPS tool scans the system from which the request is being made if it is the user of the system. If it is an SA who is requesting another system to be scanned, the tool verifies that he or she is, in fact, responsible for that computer system using a Domain Name System (DNS) query to a database containing this information. Once the system is scanned, the results are then emailed to the SA and to the user of the computer system. Figure 1 shows the process for requesting such a scan.

The SPS has a three level severity rating system. The severity rating is divided into three color-coded sections, much like a traffic light. A green color signifies ports that are usually safe to have open and used by well-known services. A yellow color identifies those ports that are cautionary ports (i.e. services that should only be run if absolutely necessary or are unknown as to what service is using it). A red color identifies those ports that are commonly associated with use by hackers and backdoor programs for unauthorized access to a system and which should be scrutinized more carefully by SAs and users.

There is also a backend interface for the administrators of the SPS tool to use in setting and updating the baselines used for comparisons by computer system and operating system. This backend is similar to many of the content management systems used on the web. Thus, an SA who has identified ports which the tool indicates as potentially unsafe (yellow), could, for a particular computer system and operating system mark it as safe, so that subsequent scans will treat the identified port as green.

To protect the entire system from unauthorized access, authentication using encrypted user identification and passwords is required. The system provides authentication via the Lightweight Directory Access Protocol (LDAP), passing credentials to a Kerberos server. The returned Kerberos credentials are then validated by the tool against an internal file, thus ensuring that passwords are not passed over the network, preventing password sniffing attacks while ensuring that only authorized users have access to the SPS tool.



Scan Life-Cycle

Figure 1: Self-Port Scanning Tool Process Flow

3. Security of the SPS Tool

A security assessment tool that is not secure itself is much like building a house with wood that is already infested with termites. In the end it will cause more headaches than it solves and will simply require much more work to be done to fix the problem. Security considerations have been a primary concern throughout the development process of the SPS tool.

If the SPS tool were compromised, this tool would change from a tool of protection into a potential attack vector by providing information on what systems may be vulnerable. Thus, protection from usages of this tool in ways that were not intended has been built into it from its inception.

One protection is that this tool only allows scans to be run against computers that are within specified subnets, thus circumventing the capability of someone using this tool against a computer or group of computers against outside the domain. In addition, the SPS tool does not present report results through the web interface. Instead, this tool emails the results to the cognizant user(s) of the computer system(s) being scanned and the cognizant SAs (as identified by the Domain Name Service (DNS) server or another specified internal process). These two features prevent a person from scanning systems indiscriminately to search for vulnerabilities that can be exploited.

A further security protection is the web server itself on which the SPS tool runs. It is critical to have a highly secure and well-monitored web server for running the tool. Locking down the web server for such a tool is essential to protect the tool and its service from being modified and used as an attack tool.

The tool was developed using Perl and a SQL database. During and following the development process the code was verified to ensure that identified unsafe libraries and routines were not used. Further protections were implemented and the code verified so as to circumvent vulnerabilities and unwanted exposures in the tool code and user interface which otherwise might be exploited. Some of these vulnerabilities that were verified include common attacks against the Common Gateway Interface (CGI). This includes more advanced attacks such as the Poison Null Byte [3], shell meta-character insertion [3], and Structured Query Language (SQL) injections as well as less technical attacks such as providing invalid data through the Uniform Resource Locator (URL).

The first of these four types is an attack that allows for the writing of a file with a filename that may be different than the one intended. Say, for example, you are writing to a file where the filename is variable but the extension is hard coded into the application. Under normal operation, the user would not be able to control the extension of the filename, but if a null byte (x00) is inserted into the filename, the shell will interpret this to mean that the filename has been terminated and thus will ignore anything that follows. This attack has been mitigated by

removing all null characters in any input that is received by the program.

The second attack involves passing special characters that are recognized by the command line interface or shell to be used for purposes of controlling the flow of data, command execution, etcetera. A list of common shell metacharacters can be found in the WWW Security FAQ [4]. Usage of shell metacharacters has been eliminated from the input into this tool by only allowing the characters that were necessary for each input (i.e. an IP Address will only contain up to three numbers followed by a '.', and so on).

The third attack against CGI scripts is the SQL injection. Though this attack isn't actually an attack against CGI scripts in particular, it does apply to this research since SQL statements are used throughout the software. This attack goes after vulnerable SQL statements and is used to reveal more information than is normally available. For example, a SQL statement such as 'SELECT * FROM database WHERE name = \$VAR'. In this statement, we are selecting all the fields in each record where the variable name equals the contents of \$VAR. The vulnerability in this method is if \$VAR contains more than just a name. If \$VAR contained 'somename OR 1=1' then every record in the database would be selected because even if name did not equal some name, one would always equal one. Protection from this attack is built into the Perl Database Interface (DBI), instead of giving the actual variable name while preparing the SQL statement, a '?' is put in its place. This '?' acts as a placeholder for one item and only one item to go into, thus protecting against SQL injections.

In addition to protecting against these three different types of attacks against CGI scripts, protection against one more less complicated attack needed to be provided. Data that needs to be preserved between instances needs to be stored in a local session file, a cookie that is sent to the user, or be passed back into the program via input through the URL. The data passed between much of the user interface in this tool is passed via the third method which created an attack vector for tainting the data that is processed by the program. Through normal use of the interface, the manipulation of hidden values in the page is impossible, but with a specially crafted URL being sent into the program, the email address that the report was sent to was found to be vulnerable to such an attack. This attack would allow for the use of the tool to arbitrarily scan any of the computers on the JPL campus. Protection from such attack was achieved by a simple check with the DNS

records for the Internet Protocol (IP) address of the client that is connecting. If the email for the cognizant user and cognizant admin supplied by the URL does not match that from the DNS records, the email address from the DNS records is used.

4. Interface for the SPS Tool

A previously developed tool that allows users and SAs to scan their computer for vulnerabilities using Internet Security System's (ISS) Internet Scanner was developed to help SAs and users to secure their systems [5]. This ISS Self-Scanning tool was written in the Perl scripting language. For consistency and to aid SAs and users of the SPS tool, its graphical user interface (GUI) was also written in Perl. The SPS tool thus acts as a counterpart to the ISS self-scan tool. In the design of the SPS tool's interface, the existing Self Vulnerability Scanner's interface was used as a model to give the SPS tool a familiar "look and feel" to make the learning curve easier. Very little additional learning is required on the part of the users of the SPS tool. Having the same look and feel also aids the users in understanding the results and the measures that can be taken to remediate potential problems. This approach is another security risk mitigation factor that can aid in increasing the security posture of the enterprise.

5. Robust Operation

Any tool that interfaces directly with users should be able to respond to unexpected input without running into problems. This is accomplished in the interface for the SPS tool by only giving the user the option of clicking on a single button to continue through the scan starting process. No input is expected to be received from the user and thus this program does not suffer from malformed user input. All input that is received through the URL is verified with trusted services such as Secure LDAP, and IP packet information. As previously mentioned in the security section, when incorrect information is received through the URL, the software will simply default to the information that is received when a query based on the IP address is received.

6. Related Research

Many tools which perform similar functions can be found throughout the web, though none provide

both the ease of use and reporting capabilities provided by the SPS tool. Bilbo [6], an automated nmap-scanner and reporter tool, allows for the scanning of multiple hosts and comparing the results to a database of previously stored results but seems to be more of a tool that a SA would use in monitoring various systems given that it lacks a user interface. Remote Nmap[7], a tool written in Python to allow for the remote control of scans using nmap, lacks a robust operational interface that requires little interaction on the side of the user and also lacks the comparison ability needed to allow the user or SA to quickly verify results.

After the completion of the tool, a Perl module named Nmap::Parser [8] was found on the web. This module could have been used inside the SPS to parse the output from nmap. The actual code in the tool that parsed nmap output was at most twenty lines and thus would not have reduced the size of the program nor increased its performance.

7. Conclusion

Though the SPS tool can become an important asset in the securing of computer systems, it is by no means a complete solution to computer security. It simply provides some diagnostic information to aid the user, SA, and security professional in identifying potential security risks on computer systems and the capability to secure further their computing resources, much like the diagnostic readout that car computers aid the mechanics job of repairing a car. The true cure to the problem of computer security lies in a combination of audited software and education; education of the administrators of the computers and the users of these systems.

The association of providing three-level severity rating for open ports and a list of services that may use those ports, including Trojan programs that may be commonly associated with an open port will help in securing systems and finding potentially unsafe services and vulnerabilities not otherwise identified in other types of vulnerability scans. Likewise, providing a secure interface to the SPS system prevents it from being mis-used either inadvertently or purposely.

All too often the real vulnerability in any system lies in the human aspect of the security chain. This tool, however, does provide a significant advantage to those wishing to protect their research and assets from compromise.

8. Acknowledgements

The research described in this paper is being carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

9. References

- [1] Samba – opening windows to a wider world, available at <http://www.samba.org>
- [2] Nmap – Free Security Scanner For Network Exploration & Security Audits, available at <http://www.insecure.org/nmap/>
- [3] Rain Forest Puppy, “Perl CGI Problems”, Phrack Magazine, Vol. 9, Issue 55, September 9, 1999, <http://www.phrack.org>
- [4] Stein, Lincoln D. & Stewart, John N., “The World Wide Web Security FAQ”, Version 3.1.2, February 4, 2002, <http://www.w3.org/Security/Faq/www-security-faq.html>
- [5] Internet Security Systems – Internet Scanner, http://www.iss.net/products_services/enterprise_protection/vulnerability_assessment/scanner_internet.php
- [6] Welkom to Bilbo, the automated scanner, <http://doornenburg.homelinux.net/scripts/bilbo/index.html>
- [7] Remote nmap, <http://nmap.sourceforge.net/>
- [8] search.cpan.org: Nmap::Parser - parse nmap scan data with perl, <http://search.cpan.org/%7Eapersaud/Nmap-Parser/Parser.pm>