A.V. Ostroukh // International Journal of Advanced Studies 2, № 3 (2012). Online access: http://journal-s.org/index.php/ijas/article/view/201236/pdf_13 (connection date 27.09.2013).

The work is submitted to the International Scientific Conference «New technologies in education», Indonesia (Bali), February, 17-25, 2014, came to the editorial office on 12.12.2013.

SECURITY SCANNERS

Shaikhanova A.K., Zhangisina G.D., Zholymbet B., Katasheva Z.C., Bolathan L.

Kazakh National Technical University named after K.I. Satpayev, Almaty, e-mail: igul7@mail.ru, gul_zhd@mail.ru

The directions in the field of information security, as an adaptive network security were considered. This directions are composed of two major technologies – security analysis (security assessment) and the detection of attacks (intrusion detection). And the subject of the paper will be the first technology aforesaid.

Introduction. The network consists of channels, nodes, servers, workstations, application and system software, databases, etc. All of these components need to be evaluated for their protection effectiveness. Means tested network security analysis and look for «weak» place in it, analyze the results and based on them create various reports. In some systems, instead of «manual» intervention by the administrator, some vulnerability that found will be eliminated automatically (for example, in the System Scanner). Here are some of the problems identified by the analysis of security systems:

✓ «hatches» in the programs (back door) and programs such as «Trojan horse»;

✓ weak passwords;

✓ susceptibility to penetration of unprotected systems;

✓ improperly configured firewalls, Web – servers and databases;

✓ etc.

Technology of security analysis is an effective method of implementing network security policies before implementing its attempt to breach the inside or outside of the organization.

The modalities of the work

There are two basic mechanisms by which the scanner checks for vulnerabilities – Scan (scan) and probing (probe) [1].

Scanning – the mechanism of passive analysis, in which the scanner is trying to determine the presence of vulnerabilities without actual confirmation of its presence – on circumstantial evidence. This method is fast and simple to implement. In terms of ISS, this method is called «inference» (inference). According to Cisco this process identifies open ports found on every network device and collects associated with ports headers (banner), found by scanning each port. Each received header is compared with table rules of network devices, operating systems and potential vulnerabilities. On the basis of this comparison are made the conclusion about the presence or absence of vulnerabilities.

Probing – active mechanism analysis, which ensures presence or absence vulnerability on the analyzed node. Probing performed by simulating the attack, using the validated vulnerability. This method is slower than the «scan», but almost always much more accurate. In terms of ISS, this method is called «confirmation» (verification). According to Cisco's process uses information obtained during the scanning process («inference»), for a detailed analysis of each network device. This process also uses well-known methods of the attacks in order to fully confirm the alleged vulnerability and discover other vulnerabilities that cannot be detected by passive methods, such as susceptibility to attacks such as «denial of service».

In practice, these mechanisms are implemented by several following methods.

«Checking the headlines» (banner check).

This mechanism is a series of tests such as «Scan» and allows you to make a conclusion about the vulnerability of relying on the information in the request header scanner. A typical example of such a test – analysis of program headers Sendmail or FTP-server that allows you to find out their version and use that information to draw a conclusion about the presence of these vulnerabilities.

«Active probing test» (active probing check).

Also related to the mechanism of «scanning». However, they are not based on checking the software version in the headlines and on the comparison of the «digital snapshot» (fingerprint) piece of software with a cast of well-known vulnerabilities. Likewise as antiviral system, comparing the scanned fragments software virus signatures that are stored in a dedicated database. A variation of this method are the check sums or the date of scanning software, which are implemented in scanners running on the operating system level.

«Imitation of attacks» (exploit check).

These checks include the mechanism of «probing» and is based on the exploitation of various defects in the software.

Some vulnerabilities do not reveal themselves until you «push» them. For that purpose against a suspect or service node they run a real attack. Header checks carried out initial inspection of the network, and the method of «exploit check», rejecting the information in the headers to simulate a real attack, thereby more effectively (but less speedy) detecting vulnerability scanning nodes. Imitation of attacks is a more reliable method of analysis of security than the header checks, and usually more reliable than active probing test [2].

However, there are cases where the simulated attack cannot always be realized. Such cases can be divided into two categories: a situation in which the test results in a «denial of service» of the analyzed

host or network, and the situations in which a vulnerability in principle, is suitable for the implementation of network attacks.

The scanning steps

Almost any scanner analyzes the security in several stages:

1. Collecting the network information. At this stage identified all active devices on your network and determined by running them services and daemons. In the case of systems security analysis at the level of the operating system, this step is skipped, since at each node of the analyzed system are set to relevant agents scanner.

2. Detection of potential vulnerabilities. The scanner uses the above database to compare the data collected from known vulnerabilities by checking the headers or active probing inspections. In some systems, all vulnerabilities are ranked according to the degree of risk. For example, in NetSonar vulnerabilities are divided into two classes: local and network vulnerability. Network vulnerability (for example, acting on routers) is considered more serious than vulnerabilities unique to workstations. Similarly, «comes» and Internet Scanner. All of the vulnerabilities in it are divided into three levels of risk: High, Medium and Low.

3. Confirmation of selected vulnerabilities. The scanner uses special methods and models (mimics) certain attacks to confirm the existence of vulnerabilities on the selected nodes of the network.

4. Report generation. Based on the collected data, the system creates a security analysis reports describing discovered vulnerabilities. In some systems (eg, Internet Scanner and NetSonar) reports are generated for different types of users, ranging from network administrators and ending with the leadership of the company. If the first is primarily interested in the technical details, it is necessary to guide the company to present a beautifully decorated with the use of graphs and charts reports with a minimum of detail. An important aspect is the presence of recommendations to address the identified problems. And here on the right is the leader of the system Internet Scanner, which for each vulnerability contains step by step instructions to resolve the vulnerabilities that are specific to each operating system. In many cases, the reports also contain links to the FTP-server or Web-based, containing patches and hot fixes, resolves vulnerability.

5. Automatic elimination of vulnerabilities. This stage is very rarely realized in network scanners, but is widely used in the system scanners (eg, System Scanner). Furthermore, this feature can be implemented in different ways. For example, the System Scanner, a special script (fix script), which the administrator can start to address the vulnerability. Along with the creation of this scenario is created and the second scenario, cancelling the changes. This is necessary if the problem is corrected; the normal functioning of the assembly had been violated. In other systems, the possibility of «rolling back» does not exist.

In any case, the administrator performing the search for vulnerabilities, there are several options for using the system security analysis:

- Start scan only for checking potential vulnerabilities (stages 1, 2 and 4). This gives a preliminary acquaintance with the systems in the network. This method is much less disruptive than others and also is the fastest.
- Start Scan for checking potential and confirmed vulnerability. This method can cause a disruption of the network nodes during the execution of audits type «exploit check».
- Start scanning with your custom rules for finding a particular problem.
- All the aforementioned

A single database format of vulnerabilities

In order to standardize and possible integration of security analysis is currently underway to create a common format for all scanners database vulnerabilities. Although this work has only just begun and it is far from being completed, the first steps have already been taken . For example, COAST Laboratory at Purdue University has developed a draft of such a database. One of the problems encountered by the researchers – a description of vulnerabilities and their controls (attacks) [4].

Languages of the description vulnerabilities and checks

Attempts to add mechanisms for describing vulnerabilities and checks in the system security analysis were carried out for a long time. They were made by almost all software companies. The first such attempt was made by Wits Venema and Dan Farmer – developers of SATAN. Description of new vulnerabilities or rather their checks carried out by means of language Perl. This is a rather trivial task required extensive knowledge of language Perl, and the architecture of the protocol stack TCP / IP and scan the operating system. The same path (using Perl) system developers went Web Trends Security Analyzer. Annex 1 provides an example of checks to determine the type of operating system being scanned host. Language Perl, along with the language C, and is used in the Internet Scanner. Moreover, in addition to features built into the system Internet Scanner, ISS delivers a separate company description system attacks APX (Advanced Packets eXchange).

Conclusion

Use of such funds is necessary. But I want to note that we should not regard them as a panacea for all ills. They do not in any way replace the security specialists. They just automate their work in helping to quickly check hundreds of nodes, including and those on other sites. They will help to detect virtually all known vulnerabilities and recommend measures to eliminate them. They automate the process, and with the ability to describe their own checks, will help to effectively apply them to any

organization's network, taking into account your particular specificity. We must remember that the scanner – it's just a part of the effective network security policy, which consists not only of the use of various technical measures of protection (security analysis tools, intrusion detection systems, firewalls, etc.), but also the use of various organizational and legislative measures.

Appendix 1.

Example audit carried system Web Trends Security Analyzer

<TestAuthor> WebTrends Corporation </TestAuthor>
<TestCopyright> Copyright 1998, WebTrends Corporation, All Rights Reserved. </TestCopyright>
<TestVersion> 2.0 </TestVersion>
==========
<TestDependency> estabvc </TestDependency>
<TestCategory> inventory </TestCategory>
==========
<TestTitle> Query OS Type via Netbios </TestTitle>
<TestVulnerabilityDescription>
This test attempts to determine the operating system type and version running on
the specified hosts.
</TestVulnerabilityDescription>
==========
<Test>
osdetectnt.pl
attempt to detect OS using a netbios over tcp/ip call
require "crowbar.pl";
$theTargetNetbiosName=GetStringParam($crowbar::WTDB_NetbiosName);
crowbar::WTDebugOutput("OSDetect -- the target netbios name is $theTargetNetbiosName");
if($theTargetNetbiosName){$a=crowbar::WTGetNTOSInfo($theTargetNetbiosName);
if($a){$a=~ /^OSTYPE (.*):VERSION (.*)/;$type=$1;$version=$2;
crowbar::WTDebugOutput("Type is $type, version is $version\n");if($version=~ m/OSVersion_Unknown/){crowbar::WTAddRecord($crowbar::WTDB_OSVersion, length("Unknown") + 1, "Unknown", -1);} elsif($version=~ m/OSVersion_WindowsNT_3_5_0/){crowbar::WTAddRecord($crowbar::WTDB_OSVersion, length("Version 3.5") + 1, "Version 3.5", -1);}elsif($version=~ m/OSVersion_WindowsNT_3_5_1/){crowbar::WTAddRecord($crowbar::WTDB_OSVersion, length("Version 3.51") + 1, "Version 3.51", -1);}elsif($version=~ m/OSVersion_WindowsNT_4_0/){crowbar::WTAddRecord($crowbar::WTDB_OSVersion, length("Version 4.0") + 1, "Version 4.0", -1);}elsif($version=~ m/OSVersion_WindowsNT_5_0/){crowbar::WTAddRecord($crowbar::WTDB_OSVersion, length("Version 5.0") + 1, "Version 5.0", -1);}if($type=~ m/OSType_Unknown/){crowbar::WTAddRecord($crowbar::WTDB_OSType, length("Unknown") + 1, "Unknown", -1);}elsif($type=~ m/OSType_Unix/){crowbar::WTAddRecord($crowbar::WTDB_OSType, length("Unix Server") + 1, "Unix Server", -1);}elsif($type=~ m/OSType_WindowsNTServer/){crowbar::WTAddRecord($crowbar::WTDB_OSType, length("Windows NT Server") + 1, "Windows NT Server", -1);}elsif($type=~ m/OSType_WindowsNTPDC/){crowbar::WTAddRecord($crowbar::WTDB_OSType, length("Windows NT Primary Domain Controller") + 1, "Windows NT Primary Domain Controller", -1);}elsif($type=~ m/OSType_WindowsNTBDC/){crowbar::WTAddRecord($crowbar::WTDB_OSType, length("Windows NT Backup Domain Controller") + 1, "Windows NT Backup Domain Controller", -1);}elsif($type=~ m/OSType_WindowsNTWorkstation/){crowbar::WTAddRecord($crowbar::WTDB_OSType, length("Windows NT Workstation") + 1, "Windows NT Workstation", -1);}elsif($type=~ m/OSType_WindowsNT/){crowbar::WTAddRecord($crowbar::WTDB_OSType, length("Windows NT") + 1, "Windows NT", -1);}elsif($type=~ m/OSType_Windows95/){crowbar::WTAddRecord($crowbar::WTDB_OSType, length("Windows 95/98") + 1, "Windows 95/98", -1); }elsif($type=~ m/OSType_Windows98/){crowbar::WTAddRecord($crowbar::WTDB_OSType, length("Windows 98") + 1, "Windows 98", -1);}}} </Test>

Appendix 2.

Example audit carried CyberCop CASL system

```
# spoof_check.cape
# this script is used by the built-in filter checks
# please do not modify it
ip
ip_version=4
ip_proto=IPPROTO_UDP
ip_flags=0
ip_id=42
ip_done
udp
udp_sport=6834
udp_dport=5574
udp_done
data=SAS-ipspoofing
end_of_packet
```

References

1. Shangin V.G. Protection of information in computer systems and networks. – DMK-Press, 2012. – 192–201 p.

2. Domarev V.V. Safety of information technology. The systems approach. – K.: TID Dia Software Ltd., 2004. – 90–92 p.

3. Zegzhda D.P. Fundamentals of Information Systems Security / D.P. Zegzhda, A.M. Ivashko – Moscow Hotline – Telecom, 2000. – 325–356 p.

4. Smoked A.P., Zefirov S.L., Golovanov V.B. Information security audit. – BDC-press, 2006. – 289–304 p.

5. Stoling William. Cryptography and network security: Principles and Practice, 2nd ed.: Trans. from English. – M.: Publishing house «Williams», 2001. – 329–372 p.

6. Torokina A.A. Engineering and technical protection of information. – Publisher «Helios ART», 2005. – 63–78 p.

7. Galatenko V.A Risk management: a review of commonly used approaches [electronic resource]. – 2012. – URL: http:// http://citforum.ru/security/articles/risk_management/ (date of circulation: 04.09.2013).

The work is submitted to the International Scientific Conference «Modern science technology», Spain, Tenerife, November, 22-29, 2013, came to the editorial office on 10.10.2013.