Passive Vulnerability Scanning Introduction to NeVO

August 2003 – revision 9

Renaud Deraison
Director of Research

Ron Gula
Chief Technology Officer

Todd Hayton
Senior Security Engineer

Introduction

Passive vulnerability scanning is the process of monitoring network traffic at the packet layer to determine topology, services and vulnerabilities. This document will discuss the technology of passive vulnerability scanning, its deployment issues and its many applications. It will also compare passive vulnerability scanning technology to network intrusion detection technology. Example 'signatures' used to detect network vulnerabilities is also included. This paper assumes the reader has a basic knowledge of TCP/IP networking, network intrusion detection and vulnerability scanning.

Tenable offers the NeVO (http://www.tenablesecurity.com/nevo.html) passive vulnerability scanner. This paper not only serves as an introduction to passive vulnerability scanning, it also includes many examples specific to NeVO.

Why Passive Vulnerability Scanning (PVS)?

Technical Limits of 'Active' Scanners

All network vulnerability scanners need to send packets or communicate in some manner with the systems they are auditing. Because of this, they are bound by the physical limitations of the networks and systems they are auditing to send and receive these packets. In other words, scanning can take a long time, especially for large networks.

In some rare cases, the act of probing may cause instability in the audited system. Network devices such as routers and switches may also be affected by large numbers of port scans, host enumeration and vulnerability testing.

And finally, networks change all too often. As soon as an active scans is finished, it slowly becomes out of date as the network changes.

Political Limits of 'Active' Scanners

Network vulnerability scanners may also have a political stigma within large organizations. For a variety of reasons, a system administrator may feel that there is no need to have a 3rd party scan their systems. To compensate for this, a passive vulnerability scanner can be deployed in a manner to watch these 'off limits' networks and report on their vulnerabilities.

Example Scenario

Consider the following situation – a security team has hired an outside firm to conduct a security audit of their perimeter network. They scan with NMAP (http://www.nmap.org), Nessus (http://www.nessus.org) and ISS Scanner (http://www.iss.net) and have a good handle on their vulnerabilities. They feel they are prepared for the audit.

The night before the outside scan starts, someone in the company places an unpatched Windows 2000 server in the DMZ. Of course this violates the company's change control procedures, but the infraction isn't caught until the outside firm starts their testing and compromises the box. Using a vulnerability scanner 'one more time' would have caught the new box, but in this case, the security group needs to get permission to launch a scan.

If a passive vulnerability scanner such as NeVO were deployed, it would have alerted on the new Windows 2000 machine, identified it's open ports and possibly detected some of it's unpatched vulnerabilities.

NeVO Passive Vulnerability Detection Technology

Network Monitoring

NeVO monitors for client and server vulnerabilities of a specific network through direct analysis of the packet stream. It 'sniffs' the traffic much like a network IDS or protocol analyzer. In order to accomplish this, it must be deployed on a network hub, spanned port of a switch or off of a network tap.

Passive Topology and Service identification

As NeVO observes network packets, it builds a model of the active hosts on a network and their services. For example, observing a TCP port 25 SYN-ACK packet from one of the monitored hosts will cause NeVO to re-evaluate the network model. It will determine if a server on port 25 is new information and if so, update the model.

NeVO uses a variety of techniques to determine if a host is alive and what the host is. It also makes use passive operating system identification by monitoring the SYN packets which may be emitted by a system during network usage. Each operating system (Linux, Windows XP, Solaris, etc.) build their SYN packets in their own unique way and this can be used to determine some operating systems.

Simple Banner Analysis

NeVO reconstructs both sides of a network conversation and then analyzes the data for evidence of specific client or server vulnerabilities. It does this through the efficient use of signatures which program NeVO to recognize vulnerable service banners.

Unique client and servers for protocols such as HTTP, SMTP and FTP have unique strings which identify the version of the service. Most people familiar with networking readily understand that network services display their versions, but clients (such as a web browser) also do this.

Intelligent Banner Analysis

NeVO's vulnerability detection is much more than an effort to match banner signatures. Many of the more complex protocols such as DNS and SNMP require several steps and logic to determine the actual version of the underlying service or client. Because of this, NeVO has a signature language that includes multiple 'regex' styles of pattern matching. In some case, knowledge of the protocols being monitored can be used to effectively write signatures. Several example NeVO signatures are included later in this document.

NeVO is Traffic Dependent

NeVO needs to see a packet in order to make a conclusion about a vulnerability. The most vulnerable server in the world which does not talk to anyone will not be detected by NeVO. However, NeVO makes an excellent fall-back choice when active scanning is not an option. It also is used to "fill the gaps" between successive active scans.

Network Deployment

Network Choke Points

NeVO should be deployed much like a network IDS (NIDS) or packet analyzer. In some cases, it may make sense to deploy it directly on a pre-existing NIDS such as Snort (http://www.snort.org) or packet analyzer.

Performance

NeVO has a much different job than a NIDS. Given 8,000,000 web sessions, a NIDS has to consider each one at length to find just one attack. NeVO can pick one of those sessions which targets a protected server, and monitor it as much as needed. Because of this, dropped packets and CPU load are not limiting factors in the effectiveness of the NeVO. It monitors several thousand sessions per second, and will automatically lock-onto sessions it is tracking and ignore new sessions if the load is to high.

Reporting

NeVO integrates with the Nessus vulnerability scanning client and also works with the Lightning Console. When deploying NeVOs, ensure that network connectivity is available to the server from the Lightning Console. For use with the Nessus client, NeVO will generate a Nessus-compatible text report which can be read and analyzed by the Nessus client.

Tenable has designed NeVO with the ability to report the collected network information with a threshold level. The threshold is the number of observed sessions on a given port before it is reported as being active. This allows a user of NeVO to select the level of sensitivity which directly correlates to false positives and false negatives. A high threshold will only report services which have high numbers of sessions. A low threshold will report on any network session it sees. Even though NeVO is 'smart' about services which open non-standard ports such as FTP and P2P applications, there are situations where legitimate ports are opened in a temporary manner.

Asymmetric Routing

If a network has an asymmetric routing topology, the effectiveness of NeVO (any any network based session tracker) will be limited. About a third of the signatures available for NeVO will not work if both sides of the network session are not present.

False Banners and Honeypots

For network administrators that have deployed network honeypots, or have altered the default banners of their services, they will have deceived a portion of NeVO's ability to identify vulnerabilities in those systems.

Example NeVO Signatures

Basic NeVO Example

This signature illustrates the basic concepts of NeVO signature writing:

```
id=1000001
nid=11414
hs_sport=143
name=IMAP Banner
description=An IMAP server is running on this port. Its banner is :<br>
%L
risk=NONE
match=OK
match=IMAP
match=server ready
regex=^.*OK.*IMAP.*server ready
```

In this example, the following fields are used:

- *id* is a unique number assigned to this plugin
- *nid* is the Nessus ID of the corresponding Nessus NASL script
- *hs_sport* is the source port to key on if we have the high-speed mode enabled
- *name* is the name of the plugin
- *description* is a description of the problem or service
- *match* is the set of match patterns we must find in the payload of the packet before we evaluate the regular expression
- *regex* is the regular expression to apply to the packet payload.

Notice that the description contains the %L macro. If this plugin evaluates successfully then the string pattern in the payload that matched the regular expression is stored in %L and is printed out at report time.

More Complex NeVO Example

```
id=1000004
nid=10382
cve=CAN-2000-0318
bid=1144
hs_sport=143
name=Atrium Mercur Mailserver
description=The remote imap server is Mercur Mailserver 3.20. There is
a flaw in this server (present up to version 3.20.02) which allow any
authenticated user to read any file on the system. This includes other
users mailboxes, or any system file. Warning : this flaw has not been
actually checked but was deduced from the server banner
solution=There was no solution ready when this vulnerability was
written; Please contact the vendor for updates that address this
vulnerability.
risk=HIGH
match=>* OK
match=MERCUR
match=IMAP4-Server
regex=^\* OK.*MERCUR IMAP4-Server.*v3\.20\..*$
```

Notice that the first match pattern makes use of the '>' symbol. The '>' symbol indicates that the subsequent string must be at the beginning of the packet payload. Use of the '>' symbol is encouraged where possible as it is an inexpensive operation.

NeVO Network Client Detection

```
id=1000010
hs_dport=25
clientissue
name=Buffer overflow in multiple IMAP clients
description=The remote e-mail client is Mozilla 1.3 or 1.4a which is
vulnerable to a boundary condition error whereby a malicious IMAP
server may be able to crash or execute code on the client.
solution=Upgrade to either 1.3.1 or 1.4a
risk=HIGH
match=^From:
match=^To:
match=^Date:
match=^User-Agent: Mozilla
match=!^Received:
regex=^User-Agent: Mozilla/.* \(.*rv:(1\.3|1\.4a)
```

Match patterns that begin with the '^' symbol mean that at least one line in the packet payload must begin with the following pattern. Match patterns that begin with the '!' symbol indicate that the string must NOT match anything in the packet payload. In this case, the '!' and '^' symbols are combined to indicate that we should not evaluate any packet whose payload contains a line starting with the pattern “Received:”.

The '^' is more expensive to evaluate than the '>' symbol. So, while both match patterns "^<pattern>" and "><pattern>" would find <pattern> at the beginning *of a packet payload*, the use of '>' is more desirable as it is less costly. Use '^' when looking for a the occurrence of a string at the beginning of a line, but not at the beginning of the packet payload. In the latter case, use the '>' character instead.

<u>NeVO can Match “Previous” Packets</u>

NeVO allows matching on patterns in the current packet as well as patterns in the previous packet in the current session. This plugin shows how we can make use of this feature to determine if a UNIX password file is sent by a web server:

```
id=800001
name=Password file obtained by HTTP (GET)
family=Generic
sport=80
description=It seems that a Unix password file was sent by the remote
web server when the following request was made :<br>%P<br>We saw :
<br>%L</br>
pmatch=>GET /
pmatch=HTTP/1.
match=root
match=daemon
match=bin
regex=root:.*:0:0:.*:.*
```

Here we see match patterns for a root entry in a UNIX password file. We also see *pmatch* patterns that would match against a packet that makes an HTTP GET request to a web server. The match patterns apply the current packet in a session and the *pmatch* patterns apply to the packet that was captured immediately before the current one in the current session. To explain this visually, we are looking for occurrences of the following:

```
                       GET / HTTP/1.*
1) client    -------------------------> server:port 80

              Contents of password file:
              root:.*:0:0:.*:.*
2) client    <------------------------- server:port 80
```

Our match pattern would key on the contents in packet 2) and our *pmatch* pattern would key on packet 1) payload contents.

<u>NeVO Can match Binary Data</u>

NeVO also allows matching against binary patterns. Here's an example signature that makes use of binary pattern matching to detect the usage of the well known community string "public" in SNMPv1 response packets (The '#' is used to denote a comment.):

```
###
# SNMPv1 response
#
# Matches on the following:
# 0x30            - ASN.1 header
# 0x02 0x01 0x00   - (integer) (byte length) (SNMP version - 1)
# 0x04 0x06 public - (string)  (byte length) (community string -
"public")
# 0xa2            - message type - RESPONSE
# 0x02 0x01 0x00   - (integer) (byte length) (error status - 0)
# 0x02 0x01 0x00   - (integer) (byte length) (error index - 0)
###
id=1400000
udp
sport=161
name=SNMP public community string
description=The remote host is running an SNMPv1 server that uses a
well-known community string - public
bmatch=>0:30
bmatch=>2:020100
bmatch=>5:04067075626c6963a2
bmatch=020100020100
```

Binary match patterns take the following form:

bmatch=[<>[off]:]<hex>

Binary match starts at <off>'th offset of the packet or at the last <offset> of the packet, depending on the use of > (start) or < (end). <hex> is an hex string we look for.

bmatch=<:ffffffff

This will match any packet whose last four bytes are set to 0xFFFFFFFF.

bmatch=>4:41414141

This will match any packet which contains the string "AAAA" (0x41414141 in hex) starting at its fourth byte.

bmatch=123456789ABCDEF

will match any packet which contains the hex string above

<u>NeVO Supports time Dependent Signatures</u>

The last plugin example shows some more advanced features of the NeVO signature language that allows a plugin to be time dependent as well as make use of the evaluation of other plugins. The plugin shows how the PVS can detect an anonymous FTP server. The *NEXT* keyword is used to separate plugins the plugin file.

```
id=700018
nooutput
hs_sport=21
name=Anonymous FTP (login: ftp)
pmatch=^USER ftp
match=^331
NEXT #-----------------------------------------------------------------
id=700019
dependency=700018
timed-dependency=5
hs_sport=21
name=Anonymous FTP enabled
description=The remote FTP server has anonymous access enabled.
risk=LOW
pmatch=^PASS
match=^230
```

Since we are trying to detect an anonymous FTP server we are going to be looking for the following traffic pattern:

```
                             USER ftp
1) FTP client   -----------------------> FTP server

                    331 Guest login ok, ...
2) FTP client   <----------------------- FTP server

                      PASS joe@fake.com
3) FTP client   -----------------------> FTP server

                         230 Logged in
4) FTP client   <----------------------- FTP server
```

Here we can not use a single plugin to detect this entire session. So, instead we use two plugins: the first plugin looks for packets 1) and 2) and the second plugin looks for packets 3) and 4).

Looking back at the above signature we can see that plugin 700018 matches 1) and 2) in the session by keying on the patterns "USER ftp" and the 331 return code. Plugin 700019 matches on 3) and 4) by keying on the patterns "PASS" and the 230 return code.

Notice that plugin 700019 has the following field: dependency=7000018. This field indicates the plugin 700018 must first evaluate successfully before plugin 700019 may be evaluated (IE, that plugin 700019 *depends* on plugin 700018's success before it can be evaluated).

While this may seem complete, one more step is needed to complete the signature for the anonymous FTP session: We need to ensure that both plugins are actually evaluating the same FTP session. We can do this by attaching a time dependency to plugin 700019. The field time-dependency=5 indicates that plugin 700018 must have evaluated successfully in the last 5 seconds for 700019 to be evaluated. In this way we can ensure that both plugins are evaluating the same FTP session.

Applications

24x7 Monitoring

With a PVS such as NeVO, it operates continuously. If a vulnerability is observed in network traffic, it will be reported on immediately. If a new server is identified, it will be reported immediately. When configured with the Lightning Console, NeVO can be used to alert security and system administrators of network changes and new vulnerabilities.

"Zero Impact" Monitoring

With the exception of being placed on a network switch span port, deploying NeVO has no impact on the monitored network. When deploying *anything* on a switch span port, the performance of the switch is impacted.

Client Vulnerability Analysis

When monitoring traffic, NeVO can determine the version of the client of many network applications such as email, web browsing and AOL clients. This allows NeVO to analyze the host-based or client side vulnerabilities which may be present. Typically, this sort of analysis can only be accomplished with a host agent, or configuring your vulnerability scanner with system credentials so it can 'log on'.

Passive Scanning of Extranets

Since NeVO has 'zero impact' it is also (for the most part) undetectable. This means it can be configured to monitor the networks of organizations you have given access to, but may not be allowed to directly scan.

For example, your company may have recently gone through a merger and has been directed to merge corporate networks. Running NeVO may allow the detection of vulnerabilities on these new portions of your network without actively scanning them. It can also provide a 'second opinion' of a service provider or hosting company.

Application Enforcement

One of the features that NeVO has is the ability to emit a TCP ‘RESET’ packet when it observes a specific vulnerability or application. Although this may seem like a good way to keep remote hackers from communicating with vulnerable servers, it’s practical application is to limit the use of unauthorized clients, servers and specific applications.

For example, if the standard corporate web application is Netscape, NeVO can be configured to stop any other connections which make use of Microsoft Internet Explorer.

This functionality can be extended to P2P networks, where applications such as WinMX and eMule present a clear liability to corporate organizations. With NeVO, users can also write their own signatures to detect applications which can be extremely useful for keeping up with the plethora of new P2P applications and abuses.

Identification of Proxies

As part of NeVO’s own analysis of web traffic, it identifies where web proxies are located and automatically treats them as such.

Integration with Vulnerability Management Systems

NeVO integrates with the Lightning Console and Proxy, also from Tenable, to provide true 24x7 coverage of vulnerability monitoring. The topology and vulnerabilities derived by NeVO are integrated into the Lightning Console as if they were derived from a normal Nessus scanner. This feature allows an organization to conduct real active scans less often and have their vulnerability database updated in-between scans by NeVO.

Network IDS Comparison

Two Different Problems

As said before, the PVS is much different from a NIDS. The PVS is much more focused on the responses from the ‘good guys’. It spends most of its time looking at the banners presented to it from a designated set of network IP address ranges. If a session occurs and the PVS is too busy, there will likely be another session it can watch when it is does processing. The PVS can monitor several thousands of network sessions at the same time, but each session is logged until completion or the presence of a vulnerability is determined.

Unlike the PVS, a NIDS needs to make a continuous best effort when it is resource constrained. For example, all NIDS have a maximum number of sessions they can watch. What happens when this number is exceeded? Something gets dropped. When a NIDS drops information, it creates a situation for a hacker to slip an attack past it.

No need to decode hostile traffic

A NIDS must also take into account that the traffic it is monitoring can be obscured by a hostile attacker in such a way that the server will decode the attack, but the NIDS won't. For example, NIDS in the 2000-2001 timeframe would do simple pattern matches on web traffic. They would not take into account the fact that a web server would also interpret the '/' character as '%2f'. This created an avenue for an attacker to launch an attack on a server and be undetected by the NIDS

A PVS does not need to worry about this. It is not likely that network users will modify their network clients in an effort to hide the identity of the unique network client.

<u>No need for continuous Alerting and Logging</u>

For each attack a NIDS observes, it must log the attack. The amount of data logged is now determined by the attackers and cannot be predicted by the NIDS operators. Because of this, bursts in traffic or bursts of attacks can overwhelm a NIDS logging capabilities which leads to dropped attacks and slow performance.

On the other hand, the PVS does not log anything until it is asked for a report. At that time, it considers it's derived model of the network and provides it to the Nessus client or the Lightning Console. The Console can also be configured to process the reports from the PVS as often as deemed necessary and alert unique system administrators and security staff when new vulnerabilities, servers and services are discovered.

<u>Detecting Vulnerabilities with a Traditional NIDS</u>

Many NIDS include vulnerability detection when evaluating a signature. This can enhance the functionality of a NIDS and reduce some false positives. However, NIDS are not designed to look for vulnerabilities. Typically, a NIDS which is programmed to find the occurrence of a particular banner will repeat that alert over and over, for each network session containing it.

Conclusion

Passive Vulnerability Scanning is not a replacement for active vulnerability scanning, but it is an extremely useful technology. When deployed by itself, the PVS will produce very interesting information about the security profile of a monitored network, but this is no means a 'total' view of network security. When deployed jointly, both technologies will efficiently detect vulnerabilities and changes to the monitored networks..

For Further Information

Please feel free to contact either email below, or visit our web site at http://www.tenablesecurity.com.

sales@tenablesecurity.com

support@tenablesecurity.com

http://www.tenablesecurity.com/nevo.html