Global Information Assurance Certification Paper

Bryan S. Brandt
GIAC Security Essentials (GSEC) LevelOne Practical Assignment
SANS Security New Orleans 2001
31 January – 2 February 2001

Evading Passive Sniffer Detection With IDS Sensors

As Intrusion Detection (ID) technology has progressed, so too has it been increasingly considered a viable aspect of the "defense in depth" ideology. While ID may not necessarily be viewed as a definitively mature technology, there are certainly a multitude of options from which to choose. Each of the available Intrusion Detection Systems (IDS) offers a unique combination of capability, configuration options, and, of course, price.

For the purpose of this discussion, the examples will apply directly to Shadow (the **S**econdary **H**euristic **A**nalysis for **D**efensive **O**nline **W**arfare, distributed by the Naval Surface Warfare Center and available from http://www.nswc.navy.mil/ISSEC/CID/) running in the Linux environment. *Network Intrusion Detection: An Analyst's Handbook, 2nd Ed.*, cites Snort (http://www.snort.org) by Martin Roesch as having "outstripped Shadow just since December 1999" (Northcutt, *et al.* 190); however, as Shadow is tcpdump-based the concepts are fundamentally easier to illustrate.

So Where Is The Problem?

Quite simply, Intrusion Detection Systems make for fairly strategic targets when their presence is discovered during network reconnaissance. One article, published in The Institute of Electrical and Electronics Engineers (IEEE) Software Magazine, actually asserts that "[s]mart intruders who realize that an IDS has been deployed on a network they are attacking will likely attack the IDS first, disabling it or forcing it to provide false information (distracting security personnel from the actual attack in progress)." (Allen, *et al.* 47)

And why shouldn't they? After all, an IDS sensor holds great potential for the assailant who can subvert it. In addition to allowing an outsider the ability to manipulate or censor log files to conceal his or her own presence, I would assert that the sensor contains a repository of packet capture data that likely offers a wealth of information about network topologies, user accounts, and passwords.

Further, once the sensor is captured, the attacker has free and clear access to a passive packet capture device without having to arouse any unnecessary suspicion; a network scan that reveals your IDS sensor in promiscuous mode is hardly noteworthy. Similarly, there is no need for the attacker to sanitize /var/log/messages because entries that show the adapter changing into and out of promiscuous mode appear equally innocuous.

What Can I Do To Prevent This?

Perhaps the best approach to reducing this threat is to first make a cursory examination of the tool set used to identify an IDS sensor. An understanding of a tool's principle of operation is key in determining how to defend against it.

For instance, the Neped (NEtwork Promiscuous Ethernet Detector) utility, distributed ca. 1998 by the Apostols group and included in several older revisions of the Trinux toolkit (http://www.trinux.org; NOTE: more recent versions have replaced Neped with the more diverse Sentinel utility available at http://www.packetfactory.net/Projects/Sentinel), exploited a flaw in the manner in which some older (specifically 2.0.x and 2.1.x) Linux kernels responded to ARP requests. When in promiscuous mode, a box would reply to an ARP request regardless of the intended recipient. If one were to include this relatively compact (205 line) utility as part of a rootkit to be uploaded to a compromised site, Shadow in its infancy (remember, this was 1998) would likely have been running on one such affected kernel and thus the locations of the sensors would be immediately evident to an attacker.

So how would this have been combated? The easiest manner in which to avoid such detection would have been to first append the /etc/sysconfig/network-scripts/ifcfg-eth0 to include a line similar to:

```
NOARP=yes
```

Then, edit the /etc/sysconfig/network-scripts/ifup script to include a branch similar to:

```
if [ -n "$NOARP" ]; then
     ifconfig ${DEVICE} -arp
fi
```

NB that the name of variable is fairly inconsequential so long as it does not conflict with any of the predefined variables accounted for in the network configuration script, as is its value; it is simply a mechanism by which the branch is activated. Once activated, this branch disables ARP resolution entirely on the sensor, thus evading Neped detection.

Unfortunately, since the days of Neped there have been many advances in passive sniffer detection. Tools such as bind's Sentinel (cited earlier) or AntiSniff (http://www.securitysoftwaretech.com/antisniff/) developed by LØpht Heavy Industries use a battery of tests to discern the existence of packet capture devices on the network. AntiSniff 2.0, currently under development, will run on Win9x (consequently, sensor detection can be performed without a permissions structure from any accessible console), WinNT/2000, and *NIX (command line versions may be run remotely on a compromised system). Additionally, this revision "is being designed to work not only on local network segments but also across routers and switches." (S.S.T., *Technical Details* n. pag.)

The "Technical Details" page for AntiSniff 1.x breaks out the tests into several categories. It first prods the Operating System for flaws in the handling of various packet types. Packets are created to address the Linux ARP flaw examined by Neped, a similar ARP flaw in NetBSD, and a broadcast Ethernet flaw in Win9x/NT. The next round of tests forges packets to provoke a reverse DNS lookup on fictitious addresses; the premise here (simplified, of course) is that if a machine is not in promiscuous mode then it will not process a packet with a destination IP not its own and thus there will be no DNS request if a sniffer is not present. The final round of tests benchmarks the network and specific boxes at a baseline level and under duress (*i.e.*, during periods of substantial network traffic). Theoretically, a machine discarding all packets destined for IP addresses not its own at the link layer should not exhibit significantly different performance metrics during periods of forced network congestion, provided said congestion is not directly addressed to that box.

Several aspects of packet capture detection have been built into AntiSniff; however, not one is indefensible where IDS sensors are concerned. While the methodology employed by AntiSniff is by no means canonical (*i.e.*, there are other, less effective methods of sniffer detection not incorporated into AntiSniff 1.x), similar defensive tactics would apply.

Going back to our original example of Shadow running on a Linux platform, the Neped discussion has already addressed the ARP test. While this is not applicable on all Linux or BSD kernels, a little preventative medicine never hurts.

The DNS issue may be dealt with similarly by altering the Shadow configuration as follows in /usr/local/logger/sensor/start_logger.pl:

```
# Prepare the parameters to pass to the tcpdump program.

      $param =  "$PROGPAR -n -s 4096 -w - -F $FILTER";
      $param .= " 2>>$LOGDIR/tcpdump.err | $GZIPPROG > $TCPLOG4 2>/dev/null";
#
```

The addition of the '–n' parameter disables DNS resolution while the logs are being collected. An alternate solution would be deleting the /etc/resolv.conf file thereby disabling DNS resolution altogether. Once the logs have been transferred, either to the analyzer or an intermediate "safe" host, DNS information can be added into the logs on an "as needed" basis without altering the original by:

```
tcpdump -a -r tcp.2001030101 > tcp.2001030101.dns
```

Dealing With Latency Tests

The latency test is perhaps the most effective in the AntiSniff arsenal; no addressable production system will escape this test. The significant qualifier here is "addressable." The AntiSniff "Goals and Purpose" statement is pretty clear that "[i]f a machine on the

network has no IP address, no IP stack associated with any of its interfaces, or has no ability to be communicated with over the network then AntiSniff will not detect it. This is perfectly acceptable, as such a machine would not be compromised over a network in the first place." (S.S.T., *Purpose* n. pag.)

The beauty of a non-addressable IDS sensor is that it provides a virtually undetectable monitoring capability (the "virtually" caveat is included for various reasons, not the least of which is that anyone having physical access to the infrastructure may notice the "extra" box that displays the promiscuity status of an adapter on the console once an hour). A scan of the IP range will not give up the location of the sensor, and its existence is equally difficult to discern with passive sniffer detection utilities.

This configuration will require, first and foremost, that a second Ethernet adapter be installed in the sensor. The key here is that this second adapter is not to have an IP address bound to it. Unfortunately, most modern Linux distributions have a very well-meaning feature that brings adapters lacking addresses down after boot; however, this is easily remedied.

The "quick and dirty" solution would be to add the following as the last line of the 'start)' section of /etc/rc.d/init.d/network:

```
ifconfig eth1 up
```

The more elegant solution would be to modify the network scripts such that they allow for an adapter without an IP address. Linux does not create a configuration file for an address without an adapter, so:

```
cd /etc/sysconfig/network-scripts/
cp ifcfg-eth0 ifcfg-eth1
```

Having done that, edit ifcfg-eth1 and remove the IPADDR, NETMASK, NETWORK, and BROADCAST entries. Edit the remaining entries such that DEVICE=eth1, ATBOOT=yes, and BOOTPROTO=none. Finally, append an entry to ifcfg-eth1 that is similar to:

NOIPADDR=yes

Then, edit the /etc/sysconfig/network-scripts/ifup script to include a branch similar to:

```
if [ -n "$NOIPADDR" ]; then
     ifconfig ${DEVICE} up
fi
```

As in the case of the ARP example, the exact variable name is unimportant so long as there is no duplication. The preferred placement of this branch is immediately before the 'else' statement to which "regular" adapters with pre-specified IP addresses fall through.

It should be noted that this modification negates the impact of the ARP and DNS modifications, as neither applies when a machine is not addressable. In other words, this change supercedes the other solutions addressed previously.

If the sensor is on an insecure hub (*e.g.*, at the perimeter, in the DMZ, etc.) and your threat model includes only outside attacks, eth0 of this sensor can tie back into your internal network. If there is a considerable internal threat, it may be advisable to implement an isolated network for your sensors and analyzer that is non-routable from all internal and external subnets. As per usual, there is a trade-off here between usability/accessibility and security, to be determined by each individual circumstance.

For The Truly Paranoid…

It is possible to take this to yet another level. Those of us that have worked with 10Base5 Ethernet (*i.e.*, AUI or "thicknet") before may remember a method of trimming connector pins or cable conductors to create a "receive only" connection. Be careful with this; if installed incorrectly, a modified cable could result in a "transmit only" system that would prevent the collection of any data. Pins 3 and 10 of the AUI connector are responsible for establishing the transmit connection; if the connections are physically severed, it is electrically impossible for that connection to transmit data. It is worth mentioning here that with a set of complicated equations and some equally complex monitoring equipment it may be possible to determine that there is a "receive only" system operating on a network from its electrical characteristics; however, to my knowledge this is not possible without physical access.

NOTE: I would advise against applying this principle to twisted pair Ethernet cabling. After some frustration and several support calls to equipment manufacturers, you will soon discover that most twisted pair devices will not produce a link when the transmit pair is cut as there are line verification routines within most modern enterprise-grade interconnection equipment (*e.g.*, hubs, switches). Rumor has it that this configuration is possible with 10/100BaseT, although it may not be worth the potential sacrifice of your data's integrity. Several 10Base5 NICs are still commercially available, as are AUI transceiver modules for hubs (*e.g.*, the 3Com 3C1206-0).

In Conclusion

IDS sensors are potentially as valuable a resource (if not more so) as the network elements they serve to protect. Patch levels should be kept current, and routines should be implemented to reduce the volume of log data that accumulates on the sensors. Where possible, efforts should be taken to conceal the very existence of sensors as the information they contain could contribute significantly to the compromise of several additional systems. In short, recognize the IDS as an asset with significant potential for either network protection or compromise and defend it accordingly.

List of References

Allen, Julia, Alan Christie, and John McHugh. "Defending Yourself: The Role of Intrusion Detection Systems." *IEEE Software* September/October 2000, 42 – 46.

Cisco Systems, Inc. *Ethernet AUI Port Pinouts*. http://www.cisco.com/warp/public/779/smbiz/service/knowledge/pinouts/aui.htm, 1 Mar 2001.

Northcutt, Stephen and Judy Novak. *Network Intrusion Detection: An Analyst's Handbook, 2nd Ed.* Indianapolis, IA: New Riders Publishing, 2000.

savage@apostols.org. *neped.c (Source: Network Promiscuous Ethernet Detector)*. http://packetstorm.securify.com/UNIX/IDS/neped.c, 1 Mar 2001.

Security Software Technologies. *AntiSniff: Technical Details*. http://www.securitysoftwaretech.com/antisniff/tech-paper.html, 1 Mar 2001.

Security Software Technologies. *AntiSniff: Goal and Purpose*. http://www.securitysoftwaretech.com/antisniff/purpose.html, 1 Mar 2001.