NIDH - Network Intrusion Detection Hierarchy: A model for gathering attack intelligence

Craig Valli

Edith Cowan University

NIDH - Network Intrusion Detection Hierarchy
A model for gathering attack intelligence

C. Valli

School of Management Information Systems
Edith Cowan University, Australia
E-mail: c.valli@ecu.edu.au

Abstract

Internet proxy systems such as Squid exchange intelligence relevant to their function as caching proxy servers via a distributed and trusted hierarchy of machines. The required intelligence is broadcast based along the network based upon established trust relationships throughout the connected network via specific port and protocols of exchange. An intrusion detection system that incorporates this functionality for gathering attack intelligence could be a formidable foe even for the wiliest attacker.

This paper will outline a possible model for the deployment of a network/distributed network intrusion detection system utilising technologies and techniques already in existence to provide the supporting infrastructure.

Keywords: *security, intrusion, detection, firewall, attack, intelligence*

Introduction

Hackers and other perpetrators have moved and adapted rapidly to leverage the distributed, sharing network paradigm presented by the Internet to wreak havoc at ever-increasing rates and with increasing complexity (Brown, Gunderson, & Evans, 2000; CERT, 1999; Dietrich, et al 2000; Fonseca, 2001; Kent, 2000; Neumann, 1999). The changing nature of network based attacks from singular to distributed modes of attack (CERT, 1999; Dietrich et al., 2000) and the use sophisticated tools to launch these attacks underlies the need for the development of distributed defensive countermeasures. A distributed network model for the sharing and distribution of attack intelligence is one such method that the proposed Network Intrusion Detection Hierarchy (NIDH) will exploit to counteract such attacks.

Most of the tools and countermeasures deployed to defend most modern networked computer systems such as firewalls, virtual private networks, intrusion detection systems have their functionality and *modus operandii* still in a singular or digital xenophobic security paradigm. Rarely if at all do these systems share attack intelligence between hosts let alone across a trusted distributed network or hierarchy.

Firewalls are very good at denying or filtering access to a network but they are not very good at gathering attack intelligence as they simply block and discard information based on existing rulesets. Most of the existing firewall systems simply store intrusions into a logfile if so configured and rarely is this data then analysed.

Intrusion detection systems are the equivalent of a digital tripwire that detects what is termed an intrusion, breach or attack of a network computer system (McCarty, 1999;

Ranum, 1999). The common flaw is that most currently deployed systems have only a single tripwire. The one point of reference for the tripwire is normally the inbound data device, a router or ethernet card for gathering intelligence for the intrusion detection system. They also rarely communicate with other systems to exchange attack intelligence.

There are many current attempts at developing systems that use distributed trusted networks to gather attack intelligence. One of the more recent additions is the Autonomous Agents for Intrusion Detection (AAFID) (Spafford & Zamboni 2000) and this goes some way to providing distributed intrusion detection systems through the use of autonomous agents. These agents report attack intelligence back to specified transceivers for processing and is one such model that is trying to address the current problems of system isolation.

There are existing systems that analyse the log files from intrusion detection systems, firewalls, and other network countermeasures. Most of these systems are capable of detailed analysis of logfiles that produce extensive and extensible static reports. What is lacking is a mechanism for the dynamic sharing of attack intelligence although several of the existing countermeasure systems already have the ability to communicate the results of analysis via a network socket.

This paper will outline a possible model for the deployment of a network/distributed network intrusion detection system utilising technologies and techniques already in existence, to provide the supporting infrastructure. The model will leverage concepts and methods already used by the Squid proxy cache system. The model is not intended to rewrite the rule book but provide infrastructure capable of a rapid response to attacks via the sharing of attack intelligence.

THE NIDH Model

Baseline Systems

The baseline systems for the model should have firewall and intrusion detection capabilities. For the purposes of the explanation of this model we can assume that the system that is described by the author operates above the firewall and intrusion detection systems as depicted in Figure 1.

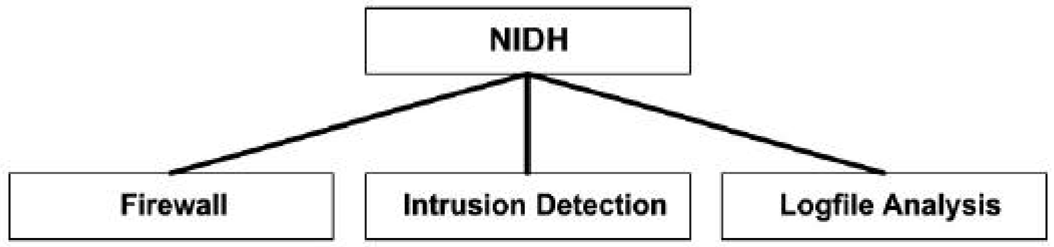

Figure 1: NIDH Architecture

The NIDH extracts information from the standard reporting functions on these systems. This information can come from a direct output stream/socket containing this information, or through the tailing of the logfiles that are files created on a disk sub-system as result of directing the output stream to disk for storage. The firewall and the intrusion detection system should be capable of at least identifying the probed or attacked port and providing this information either to the logfile or socket. The NIDH is a stand-alone process that is scalable. For example, the NIDH does not have to be located on the server that runs all of the other services, and in fact could be a separate machine that summarises this information for a complete network.

Data Analysis

The NIDH should summarise and reduce the amount of data where there is high repetition of an attack type over a short time frame. The use of an automated attack or a brute force attack would typically produce multiple logfile entries from the same attacking host in rapid succession. The following detail in the Logfile 1 exhibit is a snippet from a conventional system logfile demonstrating exactly this scenario.

```
Feb 2 12:03:04 foden netacl[28347]: deny host=adsl-63/63.202.208.126 service=wu.ftpd
Feb 2 12:03:04 foden netacl[28353]: deny host=adsl-63/63.202.208.126 service=wu.ftpd
Feb 2 12:03:04 foden netacl[28348]: deny host=adsl-63/63.202.208.126 service=wu.ftpd
Feb 2 12:03:04 foden netacl[28354]: deny host=adsl-63/63.202.208.126 service=wu.ftpd
Feb 2 12:03:04 foden netacl[28349]: deny host=adsl-63/63.202.208.126 service=wu.ftpd
Feb 2 12:03:04 foden netacl[28350]: deny host=adsl-63/63.202.208.126 service=wu.ftpd
```

Logfile 1: A sample attack or probe

The logfile format in Logfile 1 provides vital information about the attack and the attacking host. The following is an explanation of the first line of logfile entry:

- Date & Time of the attack - **Feb 2 12:03:04**
- Defensive Host - **foden**
- The process name and number responsible for the action - **netacl[28347],**
- The action taken **– deny**
- Attacking Host - **host=adsl-63/63.202.208.126 -** in the format hostname/IP Address
- Service attacked/probed **- service=wu.ftpd -** the service number for this is 21

The NIDH should summarise this amount of data to simply one reference of the attack host in IP Address format and the service in service number format being attacked for transmission through the NIDH to other systems. This reduces the amount of data that would be transmitted across the NIDH. It also takes into account and counterattacks the problems presented by the use of automated vulnerability scanners such as Nessus, SATAN, or automated scripts that are used to implement brute force attacks.

Pattern detection where possible should occur as well for instance in the case of a distributed attack. The log example Logfile 2 below has been modified to demonstrate a sample log entry for a distributed attack.

```
Feb 2 12:03:04 foden netacl[28347]: deny host=distributed_1  service=wu.ftpd
Feb 2 12:03:04 foden netacl[28353]: deny host=distributed_2  service=wu.ftpd
Feb 2 12:03:04 foden netacl[28348]: deny host=distributed_3  service=wu.ftpd
Feb 2 12:03:04 foden netacl[28354]: deny host=distributed_4  service=wu.ftpd
Feb 2 12:03:04 foden netacl[28349]: deny host=distributed_5  service=wu.ftpd
Feb 2 12:03:04 foden netacl[28350]: deny host=distributed_6  service=wu.ftpd
Feb 2 12:03:04 foden netacl[28351]: deny host=distributed_7  service=wu.ftpd
```

Logfile 2: A sample distributed attack

What differs in this case is the attacking host changes distributed_1 to distributed_7 but the attacked service wu.ftpd remains the same over a period of time. It would be highly unlikely that this behaviour or pattern is anything other than an attempted distributed attack on a system on that particular service.

Web of Control - Tripwire Web

Squid file caching systems exchange intelligence relevant to their function as caching proxy servers via a distributed and trusted hierarchy of machines. The required intelligence is broadcast based along the network based upon established trust relationships throughout the connected network via specific port and protocol of exchange.

An intrusion detection system that incorporates this functionality for gathering attack intelligence could be a formidable foe even for the wiliest attacker. The NIDH leverages this idea of a trusted hierarchy of interconnected hosts to be the backbone upon which to transfer attack intelligence between hosts to be utilised by defensive countermeasures installed on these hosts.

One of TCP/IPs inherent strengths is its ability to packet switch and route around troublespots in a network in this case a disconnected server. Due to this arrangement, a parent/sibling mesh as per Figure 2 can exist and should one system become inoperable it will not unduly affect others within the mesh.

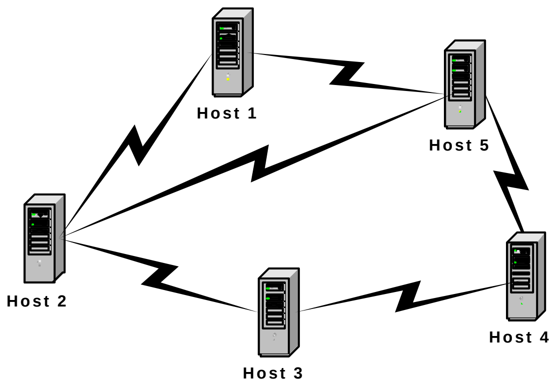

Figure 2: Sample NIDH Hierarchy

Each node on the network when arranged such as in Figure 2 having parent and sibling relationships with others becomes a trip-wire mechanism in itself. For example, if Host 3 in this scenario were attacked it would send "attack intelligence" to Host 4 who then sends to Host 5. Packets would also flow to Host 2 who sends to Host 5 and Host 1 respectively. This flow is illustrated in Figure 3

$$\begin{tikzcd}
 & & 3 \arrow[dl] \arrow[dr] & & \\
 & 4 \arrow[dl] & & 2 \arrow[dl] \arrow[dr] & \\
5 & & 5 & & 1
\end{tikzcd}$$

Figure 3: Packet Exchange Trace for an Attack originating against Host 3

So automatically, the machines that receive the intelligence packets from other NIDH connected Hosts can use this information to better defend themselves. Having received the attack intelligence packet from an upstream Parent Host, the Receiving Host can use the intelligence to protect itself. It could apply firewall routing blocks to deny the requisite services to the Attacker Host IP Address before it even begins to scan or attack that machine. For simplicity, the example given is one instance of an attack on one machine, the logic of which can apply across the entire interconnected hierarchy.

The trusting of packets that are sent between the servers in the hierarchy can be problematic if a host is compromised and used to attack the hierarchy. The firewalls and IDS on the connected systems should be set such that they only accept packets from the trusted NIDH socket and any other standard ports such as mail or web. This should be done to protect hosts against being attacked by a compromised host that is in the hierarchy. As a system that sends a scan or SYN packet to other than the trusted sockets to the other hosts will be automatically removed from the hierarchy.

The Attack Intelligence Packet

To reduce information redundancy and increase the ability of the mesh to counteract attack, the "intelligence packet" needs some special design considerations. As in Figure 3 if we were just broadcasting to the systems the Attacking Host IP 203.38.0.163 and the service/port 80 that it is attacking, then this data would become repeated. In the example given above, it would receive duplicate packets at Host 5 as it would receive this packet from both Host 4 and Host 2. So to reduce this redundancy the packet should contain a Timestamp, Victim Host, Attacking Host plus the attack detail as follows in Table 1

Time & Date 1/1/70 format	**Victim Host Details Hostname/IP Address**	**Attacking Host IP Number**	**Port Attacked**
1235663.6	Host 3/10.0.0.3	203.38.0.163	80

Table 1: Sample Attack Intelligence Packets

Hence Host 5 upon receiving the "duplicate" packet could simply drop it or compare the two packets and check the validity of the packet. Upon receipt, the packet could be passed to the firewall to block/deny access to that IP on that port based upon the firewall control policy of that particular host.

The NIDH packet may seem positively benign and lacking in information when compared to a Common Intrusion Specification Language (CISL) packet (Tung et al., 1999). The CISL is a highly detailed and rich language for the purposes of describing attacks as part of the Defense Advanced Research Projects Agency (DARPA) run Common Intrusion

Detection Framework (CIDF) project (Staniford-Chen, 1998). The intent of NIDH is to provide a fast, reactive defensive mechanism that allows rapid exchange of attack intelligence.

To analogise the difference between the two packets, consider an exchange of fire between two parties. NIDH simply identifies where the shooter and the direction the bullets are coming from. The CISL however tells you who the shooter purports to be, what bullets they are using, from how far away, what gun they are possibly using and so on, which to a large degree is superfluous information. While CISL and other rich descriptive languages may help in the ex-post analysis the basic underlying tenet is the same. If someone is spraying bullets at you, who cares what type they are. If there is an immediate threat of damage, get out of the way.

In addition, to reduce a possible broadcast storm created by small packets of information, the packets could be sent at predetermined intervals between systems in much the same way that Squid uses digests to compare cache contents between systems.

Group Memory or Digest

To make the system more effective, use of digest/memory devices will be needed. There needs to be a short-term event table, which for the purposes of this model we will call the *volatile digest,* to be held on each machine in much the same way that a bridge holds a route table. This could be stored in the random access memory of the server as a table.

Long-term memory, could be in the form of a disk written digest which the author will call the *permanent digest.* This is a similar mechanism to what Squid already uses to tell each cache what is available at its parent cache by regular exchange of digest information. The purpose or intent here is to use this permanent digest for when a server rejoins or joins the network. The server will simply request and receive a copy of the digest from its nearest neighbour system. Having got the permanent digest it would use the information to generate denial routing instructions for the firewall that is running on the system. Hence the system is using the attack intelligence stored in the collective memory of the NIDH to prevent against known attacking or malicious hosts.

Time stamping and check pointing of files should be used to send only the new additions to the digest to other computers in the NIDH. This will help overall in the reduction of traffic across a network, as the files sent will be physically smaller. It will also make it relatively easier for machines in the hierarchy to update their digests, as they will be comparing the small incoming file with the larger resident digest file as opposed to processing two large files.

Sifting the Malicious from the Non-Malicious

One of the problems with the proposed NIDH system as with all intrusion detection systems is determining what constitutes an attack. The NIDH system described here is by no means perfect in this respect. However, it is an improvement on existing systems due to the ability to gather attack intelligence from a variety of dispersed hosts.

Take for example a wily attacker who uses a single scan of a single service as opposed to a brute force attack on all ports that an inexperienced or less overt attacker would typically use. If the wily attacker replicates this attack on other hosts within the NIDH then the certainty that this attack is non-malicious declines rapidly. By utilising the volatile digest features of the NIDH and scanning it for replicated attacking IP numbers on different NIDH servers over a given timeframe, for instance 24-hours, detection of these stealthy attacks should occur. In the case of any of the intrusive IP numbers matching, then the certainty that these detected intrusions are indeed possibly malicious in intent would start to increase in probability. A rating of attack magnitude could be assigned as well for the gathered attack intelligence.

A distributed attack is problematic for most commonly available defensive mechanisms due to the singular, and often disconnected, silicon fortress mentality from which they are designed and implemented. Unlike existing systems, the NIDH by the use of distributed attack intelligence sharing should provide a method of inoculating the other NIDH connected servers against a distributed attack. The servers should have learned about the attacking machines via regular NIDH intelligence exchange.

The sometimes transient nature of attacking hosts either because of semi-permanent hosts using DHCP or the fact that potential attackers may use a variety of machines to attack can prove problematic for this countermeasure design. The proposed digest system may have some persistence problems for “legitimate” users who wish to access the system who may be using “transient IP” services for example dialup users of an ISP. However, this far outweighs the potential risk of a malicious compromise of a system. Non-malicious insiders could be problematic also but a simple trust table that ignores trusted internal IP numbers that is incorporated alongside the digest system will remove this problem.

Secure Intelligence Exchange

One of the problems with a system such as the NIDH is the secure exchange between entities in the system of intelligence information. For this system to be effective, it is important that there is some way of verifying the authenticity of the sender and the integrity of the transmitted packet. An obvious way this can be achieved is by the use of a Public Key Infrastructure (PKI) to facilitate encryption of data and also as means of authentication.

A PKI allows these attributes to be enhanced and allows for the ready encryption of data streams as well by using digital certificates. PKI also allows for the revocation of keys allowing removal of rogue or compromised servers.

Of course, the problem with a PKI is the administration and security of the actual PKI itself. One way of overcoming this is by using a network of notaries. Notaries can help administer and notarise new members into the network as well as leveraging existing Certificate Authority systems and providers such as Verisign, Thawte or other providers to further verify identity. The use of encryption and PKI while having processing overheads is necessary to increase the authenticity and integrity of the packet within the distributed network.

Conclusion

As pointed out in the introduction, the number and complexity of attacks and probes is on the increase. The use of automated tools, some of which are capable of distributed attacks, can create sophisticated and sustained attacks on hosts, with minimal computer knowledge, is also increasing. However, most of the counter measures that are in place today are still in a singular defensive paradigm and are poorly equipped to cope with these new distributed, automatic tools.

If we are to effectively reduce the risk mitigated by these increases in intensity, complexity and ubiquity of network probes and attacks, then the development of a system or systems that share attack intelligence and use distributed means to exchange this intelligence are needed. The old adage "united we stand, divided we fall" should now be, "networked we stand, distributed we fall". If we are to defeat the distributed, automaton hacker of the new millennium, distributed defensive measures must be developed.

References:

Brown, D. E., Gunderson, L. E., & Evans, M. H. (2000, Aug). Interactive analysis of computer crimes. *Computer* **33**, 69-77.

CERT. (1999). *Results of the Distributed-Systems Intruder Tools Workshop*. Pittsburgh, Pennsylvania USA: CERT® Coordination Center, Software Engineering Institute, Carnegie Mellon University.

Dietrich, S., Long, N., & Dittrich, D. (2000). *Analyzing Distributed Denial Of Service Tools:The Shaft Case*. Paper presented at the 14th Systems Administration Conference (LISA 2000), New Orleans, Louisiana, USA.

Fonseca, B. (2001, May 28, 2001). Warning: DDoS attacks on the rise. *InfoWorld,* **23**, 49.

Kent, S. (2000, Dec). On the trail of intrusion into information systems. *IEEE Spectrum* **37**, 52-56.

McCarty, R. (1999, Sep). Intrusion detection strategies and design considerations. *Sys Admin,* **8**, 57-63.

Neumann, P. G. (1999, Dec). Risks of insiders. *Communications of the ACM* **42**, 160.

Ranum, M. (1999, Fall). Intrusion detection systems: Expectations, ideals and realities. *Computer Security Journal,* **15**, 25-45.

Spafford, E. H., & Zamboni, D. (2000). Intrusion detection using autonomous agents. *Computer Networks,* **34**(4), 547-570.

Staniford-Chen, S., Tung, B., and Schnackenberg, D. (1998). *The Common Intrusion Detection Framework (CIDF).* Paper presented at the Information Survivability Workshop, Orlando FL.

Tung, B., Feiertag, R., Kahn, C., Porras, P., Schnackenberg, D., & Staniford-Chen, S. (1999, 11 June 1999). *A Common Intrusion Specification Language (CISL)*. Available: http://www.isi.edu/gost/cidf/drafts/language.txt [2001, 11th June].