

Dynamic Approaches to Thwart Adversary Intelligence Gathering

Dorene Kewley, John Lowry, Russ Fink, Mike Dean
BBN Technologies, A Verizon Company

?

Abstract

The DARPA Information Assurance Program did initial research in the area of dynamic network defense, trying to prove that dynamic network reconfiguration would inhibit an adversary's ability to gather intelligence, and thus degrade the ability to successfully launch an attack. A technique that enabled dynamic network address translation of the IP address and TCP port number combinations in packet headers was implemented in an experimental network. Two tests were conducted: one to demonstrate that it is possible to disrupt an adversary's ability to sniff network traffic effectively, and another to show that the ability of intrusion detection tools to detect an adversary can be improved. The tests were successful.

1. Introduction

The Defense Advanced Research Projects Agency (DARPA) Information Assurance (IA) Program has been focusing on researching and developing concepts of strategic cyber defense for the past three years. One of the program's grand hypotheses states the following: *Dynamic modification of defensive structure improves system assurance.* It is hypothesized that if the components of a network remain unchanged over time, adversaries will have high confidence that the intelligence they gather prior to an attack will remain valid throughout the execution of the attack. On the other hand, if the network has dynamic characteristics, the intelligence gathered by the adversary prior to an attack would be time-limited, thus inhibiting the attack.

A technique was developed under the Information Assurance Program to dynamically reassign Internet protocol (IP) address space feeding into a predesignated enclave for the purpose of confusing any would-be adversaries sniffing the network. This technique is called Dynamic Network Address Translation (DYNAT). Two experiments were run on the IA program to test this dynamic mechanism. The first experiment explored the effect of the technique on the adversary's ability to map a network, and the second experiment focused on the technique's ability to be implemented as an intrusion detection device in addition to a network address translation device. This paper presents an overview of the DYNAT technology along with the results of the two experiments.

2. Background

A series of brainstorming sessions were conducted to explore the idea of dynamic defense from a broad perspective. All strategies for dynamic defense can be characterized as configuration and posture changes that are:

- ?? conducted continuously as a normal course of action;
- ?? conducted as a result of an *internal event-based* condition; or
- ?? conducted as a result of an *external event-based* condition.

During the brainstorming sessions, it was realized that adversaries follow a *process*. The IA program primarily focuses on the well-resourced, risk-averse sophisticated adversary, such as a nation state, that attacks a network in a stealthy manner to either capture sensitive information or disrupt normal operations. Adversaries in this class do not attack a network for sport or to demonstrate prowess such as amateur hackers do. These professional adversaries have a well-defined attack process with specific goals, time constraints, and budgets. Figure 1 shows the process for a well-resourced adversary, as hypothesized during one of the brainstorming sessions. Other adversaries with different motivations and goals would have different processes.

One observation was that the adversary process has internal loops that circled through various planning stages without leading to the goal. The one that looked most fruitful was *Live Network Discovery* and we speculated that keeping the adversary stuck in that loop would cause them to consume all their resources which would lead them to give up or continue under conditions of very high risk and likelihood of failure.

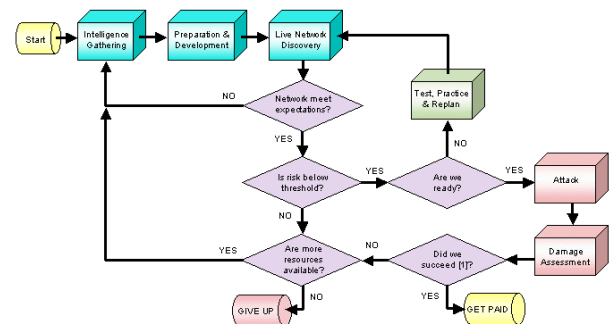


Figure 1. Proposed attack process followed by the well-resourced adversary

?

Figure 2 shows an adversary work distribution, which was hypothesized during the initial dynamic defense brainstorming session. As shown, 95% of the adversary’s time is spent preparing for the attack, while only 5% is spent actually executing the attack. These numbers were later validated in several laboratory experiments on the IA program.

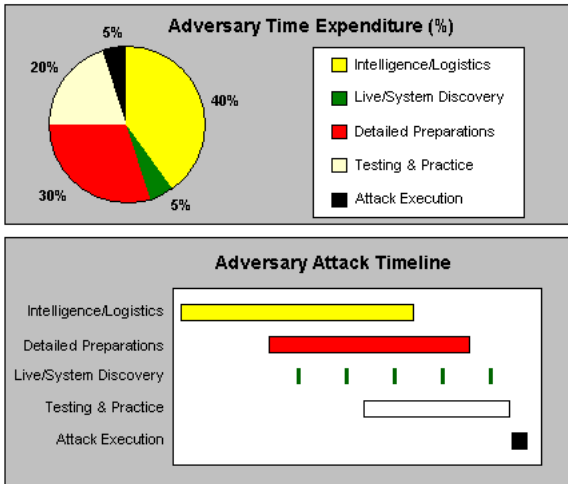


Figure 2. Adversary work distribution

Our job as defenders is to disrupt the adversaries’ processes so that they do not achieve their goals. One interesting defensive idea we stumbled upon concerns network visualization. As network defenders, we try improve our view of the network and what is happening in it. We work on improving intrusion detection systems that will catch the adversary, visualization tools that allow us to watch the network in real time, and instantaneous alerting systems. An adversary also needs to visualize the network. Wouldn’t it be interesting if we could find a way degrade the adversary’s view? This would disrupt the adversary’s process.

One way to degrade the adversary’s network view would be to introduce dynamic network modifications, which would make it harder for adversary’s to map the network. Current network attacks assume static locations of servers and services. By dynamically modifying their logical network locations, we limit the time value of intelligence gathered (sniffed) by the adversary. If an attack is not executed before the network changes again, the attacker risks failure and apprehension. By causing adversaries the repeat the intelligence gathering and development phases of the attack process, we cause the adversaries to expend additional time and resources to reach their goals. If we can keep them in this cycle, they will never attain sufficient confidence in their intelligence or their attack methodology and thus never launch their attacks.

The method of dynamic network modification discussed here is Dynamic Network Address Translation, or DYNAT. This method falls under the category of “continuously conducted dynamic configuration as a normal course of action.” Other possible methods, not

discussed here, include network address translation, (NAT) and IPsec tunneling [1].

3. DYNAT Technical Overview

DYNAT operates by obfuscating host identity information in TCP/IP packet headers when packets enter public parts of the network. Figure 3 illustrates the method. Addressing information originating from a sending client host is translated in the datagram header prior to routing to the receiving server enclave. The translation algorithm depends on a preestablished keying parameter that varies with time. The receiver, which is a server gateway in Figure 3, reverses the translation in the header fields to obtain the true host identity information. The datagram, with the original identity information, is passed into the server enclave for normal processing.

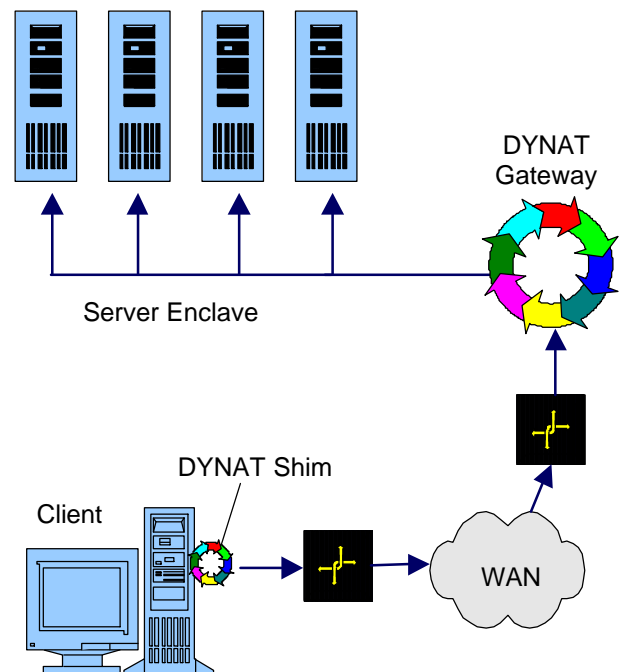


Figure 3. DYNAT approach

The host identity information could be any header information that uniquely describes a network connection between two hosts. For our experiment, the identity information consisted of (a) the host portion of the server IP address and (b) the TCP/UDP port number. This enabled us to conceal, from a layer 2 and 3 perspective, the identity of the actual server machines and services. Such concealment is sufficient to defeat a large body of network-level traffic analysis tools. The following figure, Figure 4, shows the translated TCP header parameters *in bold*.

I P	Ver.	IHL	TOS	Total Length		
	Identification			Bits	Frag Offs	
	Time to Live		Protocol	Header Checksum		
	Source Network Address			Src Host Addr		
	Dest Network Address			<i>Dest Host Addr</i>		
	Options					

T C P	Source Port		<i>Destination Port</i>		
	Sequence Number				
	Acknowledgement Number				
	Offset	Reserved	(Bits)	Window	
	Checksum			Urg Ptr	
	Options				

Figure 4. Translated parameters in TCP/IP headers

Figure 4 shows that only the host portion of the destination address is translated, not the network address. Thus, the packet can be routed normally. The number of bits that are translated is a function of the class of IP address in use. In the experiments, class B and C addresses were used. Thus, DYNAT translated the lower 16 bits in each class B destination address and the lower 8 bits in class C. The figure does not depict the UDP datagram; the destination (server-side) port of the UDP packet is translated similarly to that of TCP.

Translation is performed by a cryptographic algorithm. The client (i.e., the originating DYNAT translator host) and the server (i.e., the recipient DYNAT translator host) are configured with an initial secret value. In the experiments, a time-based mechanism was used to periodically change the secret and thus change the translation results, making it difficult for the adversary to create and maintain a map of the network. For the experiments, these methods were implemented in software as both specialized client-side TCP/IP DYNAT shim software and as a stand-alone DYNAT gateway that protected a group of servers. The client and server secret-switching mechanisms were synchronized by wall-clock time. The DYNAT software corrected potential synchronization problems by attempting to retranslate unrecoverable packets using the key of the previous time interval.

During the experiments, the Windows NT client ran a traffic generator that created realistic traffic streams via the NT TCP/IP network stack. Datagrams created by the network stack were intercepted by DYNAT shim software that translated destination information (destination IP host address and destination port), recomputed packet checksums, and forwarded the modified datagram onto the network for routing to the servers. (Two variations of the DYNAT translation algorithm were used; each variation is described in the associated configuration section below.) The server gateway received the datagrams on a public network interface, reverse-translated the destination host identity information, recomputed the checksums, and forwarded the packets to the servers on a private, server-side network interface. Responses from the servers to the client reversed this process. By translating the destination

information on datagrams sent between the client and the servers, DYNAT inhibited the adversary's ability to identify the servers and the services they provided.

4. Dynamic Network Reconfiguration Experiment

An experiment was designed to test the effectiveness of DYNAT in thwarting an adversary. The experiment focused on four primary areas:

- ?? Can dynamic network reconfiguration be effective in significantly increasing adversary work factor?
- ?? How quickly does the dynamic network reconfiguration need to be accomplished to successfully thwart an adversary?
- ?? Can an adversary recognize that dynamic reconfiguration mechanisms are in place?
- ?? Can we make an adversary change their behavior as a direct result of defender activities?

The hypothesis for this experiment was: *Dynamic network reconfiguration effectively degrades the attacker's ability to map the network, and hence increases attacker work factor and improves system assurance.* The goal of the experiment was to compare attacker work factors in the network discovery phase for static and dynamic reconfiguration network architectures. Normal network defenders are in a reactive mode. If the defenders can switch into a proactive, dynamic mode, they can potentially turn the tables on an adversary and put the adversary into a reactive mode, thus reducing the number and effectiveness of attacks.

4.1. Experimental Setup

An operational environment was built in the IA lab to model a deployed military campaign-planning unit; Figure 5 shows the network topology. By using a web browser and the Secure Socket Layer (SSL) [1] protocol, authorized deployed users were allowed to access a critical server, which was one of several servers at the home base. Authorized users could view critical planning documents containing keywords such as "weapon", "sortie", "target", and "bomb". The traffic was encrypted by SSL, so that the adversary could not simply sniff cleartext looking for the keywords. The adversary's goal was to map the defender's operational network and identify the critical servers by passively sniffing network traffic from the modeled public Internet. We assumed that once an adversary was able to identify the critical server, the adversary would then be able to attack the server at will.

The experimental network was configured to host several databases with web front ends. The databases contained maps, logistics data, intelligence data, and personnel data. Site access required authentication and was password protected. Services such as Domain Name System (DNS), e-mail, TELNET, and File Transfer Protocol (FTP) were available and running. Automated traffic generators were scripted to generate traffic and connect to the servers

on a regular basis. No intrusion detection systems were used during this experiment

A red team from Sandia National Labs acted as the adversary. As shown in Figure 5, they connected to the network from the WAN LAN, which modeled the Internet during the experiment. However, the red team approach used the IA program is not the traditional red teaming method. Because the goal of the research program is to prove or disprove hypotheses, a scientific approach is employed. Both the red and blue teams work in cooperation to conduct the experiment in a controlled fashion, and the teams are in continuous communications throughout the experiment's execution phase. This enhances the researchers' ability to collect valid experimental data to measure carefully defined metrics.

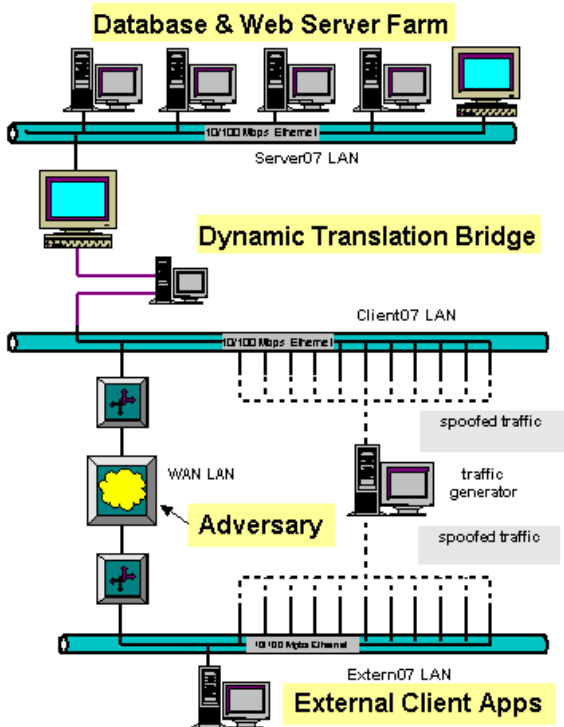


Figure 5. Dynamic network reconfiguration experiment network topology

The red team's flag (i.e., goal) was to "find the critical server." Recalling Figure 2, these red team goals represent the efforts identified as "Intelligence/Logistics" and "Live System Discovery" phases of their campaign. The information required to find this server included:

- ?? IP address and port number
- ?? Platform type (Sun or Intel)
- ?? OS version
- ?? Server type (Apache v x.x, Netscape v x.x, etc.)
- ?? LAN IP range and netmask

Because the web traffic was encrypted, the only useful information in the packets was the headers. The time for the red team to discover the above information was recorded as the primary metric.

4.2. DYNAT Configuration

For this experiment, a class-C IP addressing scheme was used. The translation algorithm used a public version of the University of California – Berkeley (UCB) crypt utility, which is a symmetric block-encryption algorithm that uses an 8-bit block size. The algorithm translates data by using a key that is derived from two independent parameters, epoch clock time (time in seconds since the UNIX epoch) and a predefined seed value. As each datagram is received from the client traffic generator, the 8 host-portion bits of the destination (server) IP address and the 16 destination TCP/UDP port bits are copied out of the datagram and packed into a 3-byte array. The bits in the array are circularly left-rotated by 4 bits, encrypted with the key, and are copied back into the datagram, overwriting the original values. The circular left-rotation step spreads the bits across normal octet boundaries, creating greater obfuscation in the encrypted result and making address/port patterns harder to discern. This is a trivial approach but adequate for the purposes of this experiment. Research is ongoing to identify better mechanisms.

After recomputing the necessary header checksums, the modified datagram is then transmitted through the network. The inverse operation is performed at the server gateway upon receipt of the modified datagram. Both the client DYNAT shim and the server DYNAT gateways were configured with a specific seed value, time intervals were determined, and the system clocks of both the client and the gateway were synchronized.

4.3. Experiment Runs and Results

Eight experimental runs were conducted using the following variations:

- ?? For runs 1, 2, and 3, DYNAT was disabled, so that the network was statically configured. This establishes a baseline to measure adversary workfactor and provides an indication of adversary learning.
- ?? For runs 4 and 5, DYNAT was enabled and the red team was given no intelligence regarding the existence or operation of the dynamic network reconfiguration mechanism. This measures the effectiveness of the mechanism against an adversary who has no prior knowledge.
- ?? For run 6, DYNAT was enabled, and the red team was given limited intelligence. They were permitted to ask questions of the blue team, but the blue team volunteered no additional information. This run measured the effect of providing partial knowledge of the mechanism to the adversary.
- ?? For run 7, DYNAT was enabled and the red team was given full intelligence, including the DYNAT code.
- ?? For run 8, DYNAT was enabled, the red team was given full intelligence, and SSL was disabled so traffic was transmitted in the clear. This was to

gain insight on the importance of data encryption versus header obfuscation.

Figure 6 presents the results of these eight runs.

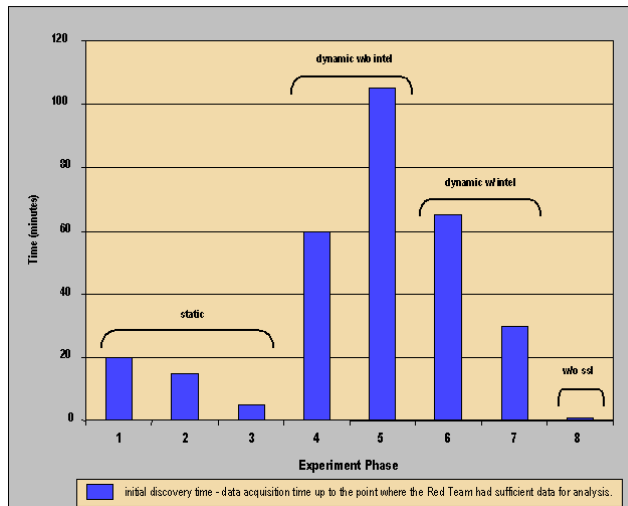


Figure 6. Dynamic network reconfiguration results

As expected, in the static configuration runs, the red team was able to quickly and accurately identify the critical server. Also, the red team learned over time, so while the first attempt took twenty minutes, the next two runs took fifteen and five minutes to complete.

In run 4, the first run with the dynamic network reconfiguration tool enabled, the red team spent an hour passively sniffing and mapping the network. They noticed that the IP addresses and port numbers were changing, but they were not able to determine what method was being used to cause this. (They believed that if they had been given more time and different tools, they would have been able to determine the method.) By tracing packet sequence numbers, they identified what they believed to be the candidate servers, although not the single primary server, and then submitted their results to the Blue Team. However, they were incorrect. They tried again in Run 5, this time spending 105 minutes gathering intelligence before completely giving up. They were confused and unable to identify even a potential list of critical servers.

The red team was then permitted to “capture” and “interrogate” one member of the Blue Team for Run 6. Some of the questions and responses were as follows:

- Q: What is being served by the servers?
A: Database text, no images.
Q: Is there a camouflage mechanism in the clients and/or servers?
A: In the clients and a proxy box for the servers.
Q: How is the change synchronized?
A: Time.

The red team then used the publicly available tool *nbtstat* to try to get the critical servers to reveal their MAC addresses. Because DYNAT was remapping the ports, *nbtstat* was not able to successfully query the servers. After

65 minutes, they gave up with no progress towards their goal.

For run 7, the red team captured a blue team member and got a full disclosure of the mechanisms involved in DYNAT through both discussions and the actual DYNAT source code. With this information, they piggybacked onto an IP/port pair that was active and reached a server. This caused the automated traffic to hang due to an artifact of the experiment. The red team did not identify or reach the critical server, but they were able to exploit the traffic to get into the network (not a goal of the experiment). The red team believed that they could have used this same technique to reach the critical server.

For the final run, Run 8, SSL was disabled, so that all traffic was transmitted in the clear. This run was added to the experiment to determine the reliance on encryption in the results. Having full knowledge of DYNAT, traffic in the clear, and a few days of learning, it took the red team only one minute to identify the critical server. Because the red team was doing passive sniffing, and their goal was to simply identify the critical server, they were able to analyze the contents of the data portion of a packet and search for keywords (such as “weapon”). Thus, we observed that encryption is a critical element of the dynamic network translation technology in that both security layers need to work together.

4.4. Experiment Conclusions

The dynamic network reconfiguration tool made the discovery efforts more difficult for the adversary. Even with extensive information on the remapping technique, the red team was not able to accurately identify the critical server in a timely fashion. The work factor increase is estimated to be *at least* in the range of 2:1 to 4:1. With these results, we can conclude that the hypothesis, “*Dynamic network reconfiguration effectively degrades the attacker’s ability to map the network, and hence increases attacker work factor and improves system assurance,*” was clearly supported.

DYNAT also changed the adversary’s behavior. An adversary simply poking around would likely not find anything interesting by looking at traffic external to the DYNAT mechanism as there were no consistent streams of data being communicated between hosts.

An interesting artifact of the experiment was the statistical “whiteness” of the automatically generated traffic. Because each packet was of similar size, there were no clues the adversary could draw from comparing packet sizes in any particular group to known expectations.

For the adversary to succeed in achieving their goals on this network, they would have been forced into a substantial data collection and analysis mode. This would cause them to stay on the network longer, thus increasing their risk of being detected. They would also be required to analyze their data using alternate tools, likely on different platforms. This movement of large files also increases their potential to be detected.

The blue team’s cost for the network reconfiguration scheme was minimal. The implementation had no impact

on the client or server applications and communications, and negligible impact on the network architecture. The DYNAT mechanism can be put in place on the wire before a front door firewall without an IP address requirement or any further network architecture impact.

The remapping rates used in this particular experiment were on two-minute and five-minute intervals. The particular rate used did not appear to impact the adversary work factor in any way, and they were able to figure out the rates. As this was a first attempt at dynamic network reconfiguration, the rate was chosen somewhat arbitrarily and simply. Future versions of the DYNAT mechanism will explore a remapping scheme need not be applied to the full IP/port range. Rather, it will be applied incrementally while keeping existing connections intact. Packet by packet remapping is also under consideration, as is randomized remapping of both the time and the IP/port ranges.

The importance of traffic encryption used in combination with the DYNAT tool should not be underestimated. If SSL encryption is disabled, the adversary is quickly able to identify the target server by simple analysis of the contents of each packet payload. In DYNAT using time windows (versus per packet translation) this gives the adversary time to do his work within that time window.

The dynamic network address translation experiment demonstrated that the tables could be turned on the adversary shifting the asymmetry in favor of the defender.

Another surprising idea that came out of this experiment: there is a strong potential for positive synergy between dynamic network reconfiguration and intrusion detection systems. If a known IP and port remapping scheme is underway, a packet that produces an unexpected translation is an immediate cause for suspicion. The following experiment explores this idea in detail.

5. Experiment Exploring DYNAT as an Intrusion Detection Mechanism

The intent of this experiment is to examine the effectiveness of synergy between dynamic network reconfiguration and intrusion detection. A static configuration of networks and systems allow sophisticated adversaries to avoid detection by hiding in or using normal network traffic and conditions. Very few intrusion detection systems will notice adversaries that look like normal traffic, and intrusion detection systems also must be tuned to ignore normal traffic to keep the false alert rate low. Dynamic network reconfiguration, on the other hand, can force or deceive the adversary into taking actions that the adversary believes to be normal, but that are actually substantially and controllably anomalous from the perspective of the system. That is, the anomaly generated by the attacker can be tailored for ease of detectability by conventional methodologies, while normal traffic does not exhibit these anomalous conditions.

Based on this concept, the experimental hypothesis explored herein is: “*Dynamic network reconfiguration can significantly improve the effectiveness of [some types of]*

intrusion detection systems and hence improve system assurance.”

If the defenders are using DYNAT to dynamically switch IP addresses and TCP port numbers, then a packet or series of packets that come into the network addressed for non-existent hosts or ports cause suspicion. Such packets might indicate malicious activity and, therefore, should cause a security alert. Statistically, DYNAT will translate intruder packets onto unused addresses or ports. All such packets become targets for analysis by the defender.

Now, suppose that an adversary sniffs traffic, somehow identifies what is believed to be the critical server, and launches an attack against this “known” target. If the defenders have simultaneously used DYNAT, to change the server address, then the defenders will see a packet arrive that is addressed to what was *previously* the critical server. This is a strong indication that hostile activity is occurring on the network, and that an adversary is using a method of network intelligence gathering and targeting.

5.1. Experimental Setup

Figure 7 shows the experimental network that was built in the IA lab. This network modeled an operational scenario where deployed forces, which needed to place orders for refresh and resupply items, used a web browser (*pringle* on the Experimental LAN) to connect to a web server that interacted with a database (*beetle* on the ia0106 LAN). Automated traffic and human generated traffic were employed. A traffic generator was also used to generate background traffic during the experiment. The LAN infrastructure used hubs, not switches.

In the previous experiment, it was important for packet payload encryption to be used in conjunction with DYNAT, to prevent the adversary from extracting information from the packet payloads. In this experiment, all traffic was transmitted in the clear, because the adversary was challenged to achieve a different goal. In this case, the adversary was challenged to *actively* disrupt normal operations; passively identifying target machines was not enough. Encryption is always recommended as a good security practice, but it was not necessary for achieving the experimental goals and was therefore not implemented.

The red team defined a broad set of attacks that were used against the network in a controlled fashion, first under static conditions, and then again with DYNAT in place. The attacks included scanning, session hijacking, and denial of service. In this experiment, the red team was not challenged to capture a flag as in more traditional red team experiments on the IA Program, rather each individual attack step was analyzed to determine if it was detected or not.

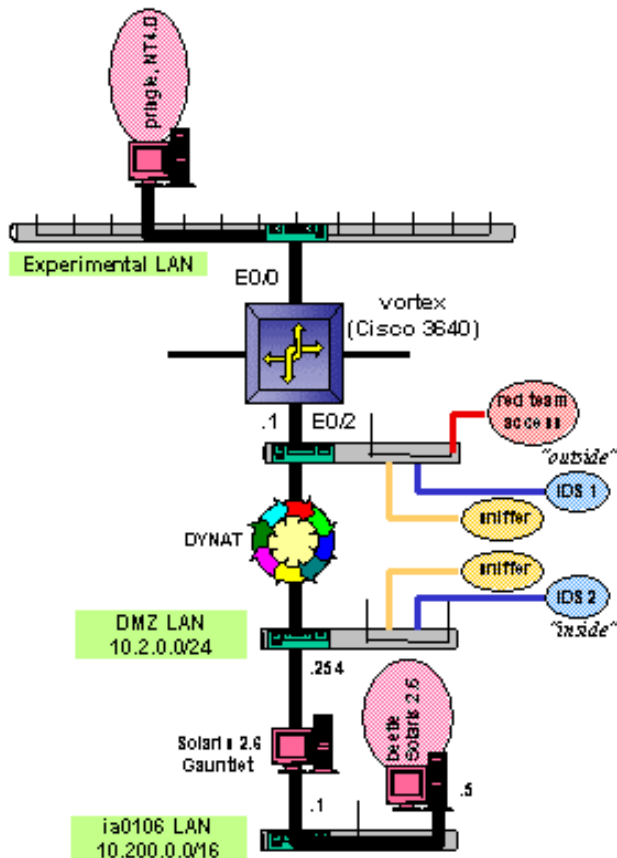


Figure 7. Dynamic network reconfiguration in conjunction with intrusion detection experiment network topology

The primary intrusion detection system (IDS) used in this experiment was a commercial off-the-shelf, network-based intrusion detection tool used in many networks today to detect a predefined set of actions known to be adversarial in nature. Two of these systems were deployed in the experimental network for comparison purposes, one on either side of the DYNAT mechanism. This is shown in Figure 7 where “IDS” represents each of the systems. The commercial ID system labeled IDS 1 was considered to be “outside” of DYNAT. The intent was to configure it using today’s security best practices for a static network in normal operations.

IDS 1

- ??Security Events – medium alert
 - o All events except netbios.
- ??Connection Events – high alert
 - o All connections from any source IP to host beetle for ftp, telnet, smtp, finger, http, pop3, portmap, ident, nntp, udp portmap, icmp echo request.
- ??Filter Events – ignore all TCP connections from any port to beetle port 80.

The IDS system on the DMZ LAN, labeled IDS 2 and sitting “inside,” was configured to best monitor a network where dynamic network reconfiguration was being employed. The two IDS systems were not expected or configured to return the same results.

IDS 2

- ??Security Events – medium alert
 - o All events except netbios.
- ??Connection Events – high alert
 - o All connections from any source IP to 10.200.0.0/16 (Class B) for any TCP, any UDP, and any ICMP echo request.
- ??Filter Events – ignore all TCP connections from any source IP to beetle for ftp, telnet, smtp, http, pop3, and UDP connections for DNS.

Besides the commercial IDS system, DYNAT itself was configured to be an intrusion detection device. Because DYNAT has full stateful knowledge of the network “behind” it, DYNAT immediately knows if a translated packet is destined for a legitimate host and service or not. In the experiment, DYNAT raised an alert for two activities:

- ?? packets attempting to use an address and port combination from a previous cycle, and
- ?? packets attempting to use invalid translated address and port combinations.

With each IDS, the detection metric is absolute; either the system detects the adversarial packet, or it does not. Commercial sniffers were also placed in front of and behind DYNAT to log traffic for a baseline measurement.

5.2. DYNAT Configuration

This second experiment used the DYNAT architecture and software from the first experiment, but with a modified translation algorithm. Also, a class-B network structure was used (16 bits for the network portion of the address, 16 bits for the host portion).

The UCB crypt algorithm, which has an 8-bit block size, was replaced with a demonstration version of the RC5 algorithm, obtained from RSA Data Security Incorporated. The use of an industry-accepted cryptographic algorithm helped to strengthen effectiveness of the DYNAT translation technique. The RC5 algorithm is a strong symmetric block cipher that uses a variable-length block size. For this experiment, RC5 was used with a block size of 32 bits. This enabled both the class-B host address and the port information to fit inside a single 32-bit encryption block.

A property of a strong symmetric block cipher such as RC5 is that each bit in the encrypted result is a function of every bit in the original plaintext. By encrypting both the IP address and the TCP/UDP ports in the same 32-bit encryption block, the translated IP address and port values become tightly coupled. As an example, a one-off difference in *either* the plaintext port *or the* address value among two datagrams would result in a significant difference in *both* the translated port *and* address values.

This implies that an adversary would have to know both the encrypted port and address values in order to "sneak a packet" into the server enclave, and even then it would only apply to the services on that specific port. This has the effect of denying broad-ranging port scans and a large set of other common attacks.

For this intrusion detection experiment, the network setup was identical to that of the experiment described in Section 4. However, the gateway software was modified to log receipt of "bad" packets, i.e., those packets which would not map onto a known server machine or port. This modification enabled the DYNAT gateway to operate as an intrusion detection sensor, highly sensitive to port scans or other unauthorized host/service-oriented activity.

5.3. Experiment Execution and Results

The red team launched three types of attacks against the network—scanning, denial of service, and session hijacking—with the following results:

5.3.1. Scans and Network Mapping. The red team began their attack set with a SYN scan of the entire Class B (10.200.0.0) address space hosting the critical server. They used a publicly available hacker tool called "nmap" to send "bare ACKs" to each address, expecting to get reset messages back from live hosts. The IP addresses are selected randomly from the Class B space. Once a reset message is received indicating a live host, nmap then begins a TCP SYN scan of the port range on the live host, again in random order. The ACK scans and SYN scans are continued in an alternating fashion.

For the static configuration, the entire Class B address space was scanned, the primary server *beetle* was located, and all live ports were identified. The entire process took fifteen minutes. Over 400 packets went through the system in this attack. IDS 1 returned the following results:

- ?? no detection of the ACK scan;
- ?? 2 port scan alerts (security events);
- ?? 10 connection attempt alerts.

For the dynamic configuration, a scan of the entire Class B address space was attempted, starting with port 80, but the live host was never identified. With DYNAT enabled, each packet header is pulled apart, and the IP and ports are reassigned to other numbers using the switching algorithm. While nmap thinks it is consistently trying port 80 over the entire Class B address space, it is actually trying random combinations of IP addresses and port numbers. It is statistically possible for one of those combinations to hit on a live host and enabled port and return a reset to nmap, but that information is time limited.

DYNAT also translates packets being returned to the "outside" network. During this attack, the nmap ACK scan by chance hit on a live IP/port combination (10.200.25.246), and then DYNAT translated the address so the reset appeared to come from 10.200.0.0 (the broadcast address). Then nmap proceeded to SYN scan the entire port range on 10.200.24.246. It spent fifteen hours looking for live ports before the attack was manually stopped. IDS 2,

which was configured to operate best under dynamic conditions, reported the following results at the end of the attack:

- ?? 1,603 SYN Flood alerts (because DYNAT was remapping IP addresses and TCP port numbers, the ID tool was seeing what appeared to be a SYN flood, although in reality it was an ACK scan)
- ?? 48,395 connection events reported instantaneously
- ?? Numerous security events reported thousands of packets into the scan

The intrusion detection features of DYNAT reported alerts for each of the red team packets. These alerts were caused by red team packets being translated to invalid IP/port combinations for the current topology or for the previous mapping scheme.

Over 1.9 million packets went through the system over the 15-hour scanning period. The IDS's GUI "froze" due to the high level of packets being processed. Furthermore, the numbers presented above actually only represent a fraction of the detections as the log files on the commercial IDS overwrites itself when filled. In this case, it was overwritten numerous times during the 15 hours. The two key results here are that the nmap tool was essentially useless to the adversary, and the adversary's scan was extremely noisy and caused the IDS to react strongly. It is safe to say that DYNAT changed the adversary's behavior in this attack.

TABLE 1: SYN scan data

	Static Network Configuration		Dynamic Network Configuration			
	Outside RealSecure		Inside RealSecure		DYNAT	
total alerts	detect	FA	detect	FA	detect	FA
	12	0	48,998	56	125,674	168
alert types	2 PortScan		48,395 Connect		125,674 UNUSUAL	
	10 Connection		1,603 SYN Flood			
packet count	total all packets: 397,125		total all packets: 130,661			
	total red packets: 393,747		total red packets: 125,674			
	total blue packets: 3,120		total blue packets: 4,120			

Table 1 presents scan data for both the static and dynamic runs. Figure 8 presents the results graphically to clarify what the IDS saw. In the static case, the IP address and port number ranges are scanned in a clear fashion. In the dynamic case, the IP/port combinations are pulled apart and translated using the time-based algorithm to random combinations all over the space. Figure 9 presents the results of what the IDS saw in a time-based graph.

5.3.2. Denial of Service Attack. The red team next launched a denial of service (DoS) attack against *beetle* to consume its CPU using the readily available hacker tool "Stream.c." This tool launches so-called "high touch" packets, which are defined as "those requiring significant handling by the target." Using a sniffer to decode the packets, it was deduced that the packets launched towards *beetle* were TCP NULL packets (no bits set in the TCP header), with the IP type of service field set to "8" (high throughput). The source IP address and port number of all packets was spoofed randomly, but all were launched against port 80 on *beetle*.

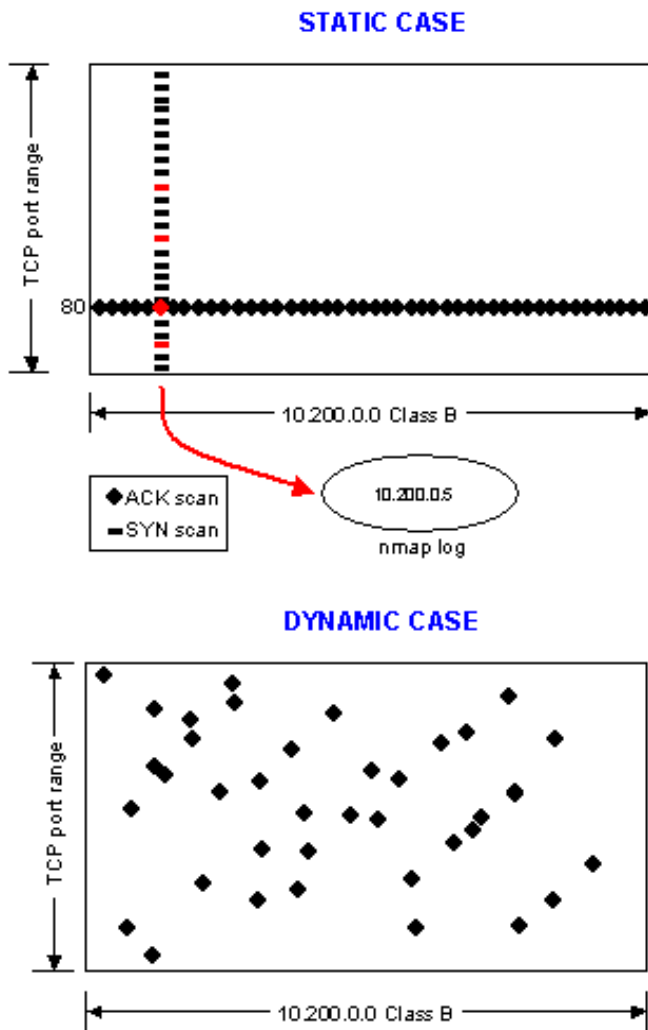


Figure 8. TCP SYN scan: static versus dynamic results of what the sensors saw in the IP/port range

In the static case, the red team located the server *beetle* and used Stream.c to launch packets against *beetle* at a rate of 5,700 packets per second. As expected, all packets “hit” the server. IDS 1 reported two alerts for each red team packet, totaling 50,000 detections. The first was a security event, and the second was a connection event. The timing of the IDS alerts lagged significantly behind the actual time of the event because the IDS system was heavily loaded by logging the activity. The IDS GUI also froze, but all packets continued to be logged to the database..

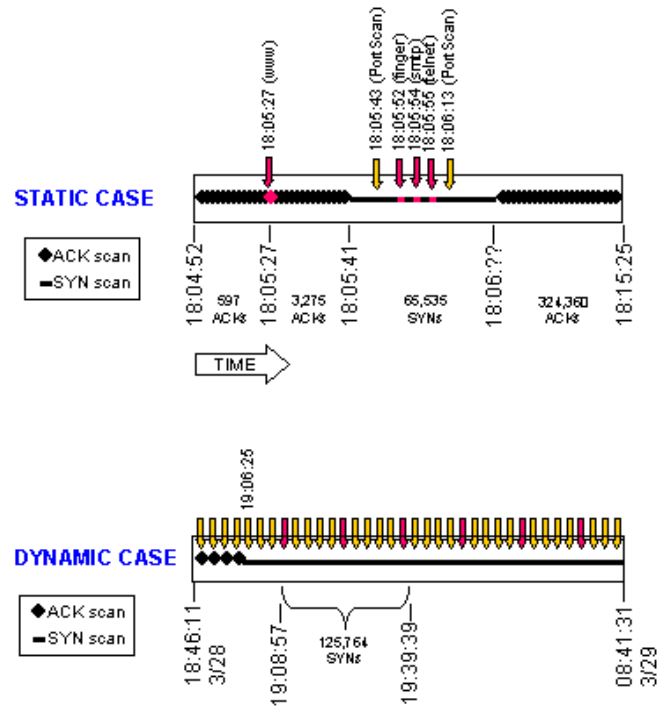


Figure 9. TCP SYN scan: static versus dynamic results of what the sensors saw over time

In the dynamic case, the red team again launched Stream.c against the critical server, *beetle*, at 5,700 packets per second. Because the red team had sniffed traffic to locate *beetle*, by the time they launched the attack, DYNAT had gone through another remapping cycle so that all packets destined for the known IP and port were remapped to a random IP/port combination. Thus, no red team packets reached *beetle*. Theoretically, if the red team had launched their attack prior to remapping occurring, those packets would have hit *beetle* during the time window before the remapping. IDS 2 again reported two alerts for each red team packet. Again, the GUI froze, but the database continued to log all alerts. The DYNAT intrusion detection tool reported one alert for each red team packet.

5.3.3 TELNET Hijacking. The red team used a program called “hunt” to launch a TELNET hijacking attack. Through traffic analysis, they learned the MAC addresses of the router *vortex* and of the Gauntlet firewall protecting the ia0106 LAN. They then sent *vortex* an unsolicited “ARP reply” showing their own MAC address corresponding with the firewall’s IP address. Since *vortex* believed this IP address was the next hop along the way to the ia0106 LAN, it sent all ensuing packets to the red team’s host. The red team was then able to modify, insert, delete, and read all packets at will. They then forwarded the packets to the firewall for delivery to *beetle*.

In the static case, the red team had unlimited time to identify and hijack the TELNET session. As expected, the IDS did not detect the attack.

In the dynamic case, the red Team was again able to identify the TELNET session and hijack it. The IDS did not detect this, and neither did the DYNAT IDS. When a remapping occurred, the hijacked connection broke, generating a "DYNAT_BOUNDARY" alert. It is hypothesized that if TCP sessions were maintained across DYNAT rollovers, the false positives would go away and the hijack would be detected. Overall, with dynamic network reconfiguration, the hijacked sessions were time limited.

5.4. Conclusions

The adversary's scans, denial of service attacks, and telnet hijacking attacks were readily detected when dynamic network configuration was used. Due to the IP/port remapping, the adversary was led to take actions that were more apparent in the normal network traffic, and the commercial IDSs were easily configured to watch for these actions. The experimental hypothesis, "*Dynamic network reconfiguration can significantly improve the effectiveness of [some types of] intrusion detection systems and hence improve system assurance,*" was in fact supported.

Further, we conclude that during the scanning attack, the dynamic network address translation tool:

- ?? eliminated the adversary's ability to successfully conduct active scanning;

- ?? made passive sniffing difficult and time limited;

- ?? enabled different intrusion detection configurations to detect many scans immediately; and

- ?? detected all attacks immediately itself.

It can be concluded that, during the denial of service attacks, DYNAT:

- ?? successfully protected the server; and

- ?? allowed only authorized traffic to connect to server.

It can be concluded that, during the session hijacking attacks, the time-based dynamic network address translation mechanism:

- ?? does not prevent hijacking, but does make it time-limited; and

- ?? can detect hijacks itself at the rollover points.

6. Dynamic Network Reconfiguration Conclusions

As stated in the introduction, one of the Information Assurance Program's grand hypotheses is, "*Dynamic modification of defensive structure improves system assurance.*" The two experiments described here show that through the use of a dynamic network address translation tool, the security assurance of a system can be increased. This is the first experiment on the IA Program to demonstrate this.

This is also the first time on the IA Program that we've been able to turn the tables on an adversary to put them in a reactive mode, as opposed to their usual confident proactive attacking mode. The methods the adversary had for gathering intelligence no longer worked. The adversary was severely time limited by the dynamic nature of the network, and forced into more vulnerable and detectable behavior. This is a significant positive result for those who are responsible for defending network assets.

BIBLIOGRAPHY

- 1) Stallings, Williams "Cryptography and Network Security, Principles and Practice" Second Edition. Pp 444-461. Prentice-Hall, New Jersey 1999.