

SAND REPORT

SAND2002-3613

Unlimited Release

Printed November, 2002

Final Report for the Network Security Mechanisms Utilizing Network Address Translation LDRD Project

John Michalski, Carrie Price, Eric Stanton, Erik Lee, CHUA, Kuan Seah,
Wong, Yip Heng and TAN, Chung Pheng

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of
Energy under Contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.



Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: <http://www.doe.gov/bridge>

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: <http://www.ntis.gov/ordering.htm>



This effort was performed as part of the Information Operations Red Team and Assessment (IORTA) program's research efforts at Sandia National Laboratories. See <http://www.sandia.gov/iorta/> for more information.



This page left intentionally blank.

SAND 2002-3613
Unlimited Release
Printed November 2002

Network Security Mechanisms Utilizing Dynamic Network Address Translation

John Michalski, Carrie Price, Eric Stanton and Erik Lee
Networked Systems Survivability & Assurance, Dept 6516
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-0785

CHUA, Kuan Seah, Wong, Yip Heng and TAN, Chung Pheng
DSO National Laboratories
Singapore

ABSTRACT

A new protocol technology is just starting to emerge from the laboratory environment. Its stated purpose is to provide an additional means in which networks, and the services that reside on them, can be protected from adversarial compromise. This report has a two-fold objective. First is to provide the reader with an overview of this emerging Dynamic Defenses technology using Dynamic Network Address Translation (Dynat). This “structure overview” is concentrated in the body of the report, and describes the important attributes of the technology. The second objective is to provide a framework that can be used to help in the classification and assessment of the different types of dynamic defense technologies along with some related capabilities and limitations. This information is primarily contained in the appendices. (*See appendix A, B and C*)

TABLE OF CONTENTS

1. INTRODUCTION	1
2. DYNAT PROCESS CONTROL	1
2.1 Time Based Synchronous.....	2
2.2 Time Based Polling.....	3
2.3 Packet or Frame Based.....	3
2.4 Key types and Distribution.....	6
3. DYNAT CODE TYPE.....	6
3.1 Software Approach.....	6
3.2 Hardware Approach.....	7
3.3 Hybrid Approach.....	8
4. ARCHITECTURE SUPPORT.....	8
4.1 LAN Segment Distant Server.....	8
4.2 LAN Segment Local Server.....	9
4.3 Local Router to Local Router.....	10
4.4 Gateway to Gateway.....	10
4.5 LAN segment to LAN segment.....	11
5. FIELD OBFUSCATION.....	12
5.1 Protocol Fields.....	12
6. PERFORMANCE DEGRADATION.....	12
6.1 Network Protocol.....	12
6.2 Application.....	14
6.3 Security.....	15
6.3.1 Gateway to Gateway.....	16
6.3.2 Local Router to Local Router.....	16
6.3.3 LAN Segment Distant Server.....	17
6.3.4 LAN Segment Local Server.....	17
6.3.5 Segment to Segment.....	17
6.3.6 Label Based Switching.....	18
6.3.6.1 MPLS.....	18
6.3.7 VLAN.....	19
6.3.7.1 802.1Q.....	19
6.3.7.2 ISL.....	20
7. IPSEC ENVIRONMENT	20
7.1 Combining Dynat and IPSec.....	21
7.2 Inter-operational Issues.....	22
7.2.1 Similar Protection Services.....	22
7.2.2 Benefits of Dynat in IPSec Environment.....	23
7.2.3 IPSec Strengthens Dynat.....	24
7.2.4 Security Implications.....	25
7.2.5 Summary.....	26
8. IDS INTEGRATION	28

REFERENCES	30
APPENDIX A	32
<i>Decision Tree</i>	<i>32</i>
APPENDIX B.....	34
1. <i>Topology A</i>	<i>34</i>
<i>Implementation Discussion</i>	<i>34</i>
A. <i>Topology</i>	<i>34</i>
B. <i>Process Control Type.....</i>	<i>34</i>
C. <i>Field Obfuscation</i>	<i>35</i>
D. <i>Key type and Distribution.....</i>	<i>35</i>
E. <i>Dynat Code Type.....</i>	<i>36</i>
2. TOPOLOGY B	37
<i>Implementation Discussion</i>	<i>37</i>
A. <i>Topology</i>	<i>37</i>
B. <i>Process Control Type.....</i>	<i>38</i>
C. <i>Field Obfuscation</i>	<i>38</i>
D. <i>Key Type and Distribution.....</i>	<i>39</i>
E. <i>Dynat Code type.....</i>	<i>40</i>
3. TOPOLOGY C	40
<i>Implementation Discussion</i>	<i>41</i>
A. <i>Topology</i>	<i>41</i>
B. <i>Process Control Type.....</i>	<i>41</i>
C. <i>Field Obfuscation</i>	<i>42</i>
D. <i>Key Type and Distribution.....</i>	<i>43</i>
E. <i>Dynat Code type.....</i>	<i>43</i>
4. TOPOLOGY D	43
<i>Implementation Discussion</i>	<i>44</i>
A. <i>Topology</i>	<i>44</i>
B. <i>Process Control Type.....</i>	<i>44</i>
C. <i>Field Obfuscation</i>	<i>45</i>
D. <i>Key Type and Distribution.....</i>	<i>45</i>
E. <i>Dynat Code type.....</i>	<i>46</i>
5. TOPOLOGY E.....	46
<i>Implementation Discussion</i>	<i>47</i>
A. <i>Topology</i>	<i>47</i>
B. <i>Process Control Type.....</i>	<i>47</i>
C. <i>Field Obfuscation</i>	<i>48</i>
D. <i>Key Type and Distribution.....</i>	<i>48</i>
E. <i>Dynat Code type.....</i>	<i>49</i>
APPENDIX C	50
1. INFORMATION ASSURANCE	50
2. ADVERSARIAL IMPACT	51
2.1 <i>Network Reconnaissance.....</i>	<i>52</i>
2.1.1 <i>Dynat Response</i>	<i>53</i>
2.2 <i>Man-In-The-Middle</i>	<i>53</i>
2.2.1 <i>Dynat Response</i>	<i>54</i>
2.3 <i>Denial of Service</i>	<i>55</i>

2.3.1 Dynat Response	59
3. TRAFFIC ANALYSIS	60
3.1 HINDERING TRAFFIC ANALYSIS	60
3.2. TRAFFIC ANALYSIS PROTECTION MODEL	62
3.3 ADVERSARY MODEL.....	66
3.4 SECURITY ASSESSMENT	66
3.4.1 <i>Information Obfuscation</i>	67
3.4.2 <i>Route Hiding</i>	70
3.4.3 <i>Limitations and Recommendations</i>	72
3.4.4 <i>Other Issues</i>	73
SUMMARY	75

ABSTRACT

A new protocol technology is just starting to emerge from the laboratory environment. Its stated purpose is to provide an additional means in which networks, and the services that reside on them, can be protected from adversarial compromise. This report has a two-fold objective. First is to provide the reader with an overview of this emerging Dynamic Defenses technology using Dynamic Network Address Translation (Dynat). This “structure overview” is concentrated in the body of the report, and describes the important attributes of the technology. The second objective is to provide a framework that can be used to help in the classification and assessment of the different types of dynamic defense technologies along with some related capabilities and limitations. This information is primarily contained in the appendices. (*See appendix A, B and C*)

DYNAT Technology Framework Review

1. Introduction

Current network security technologies concentrate on protecting networks by developing a strong perimeter defense. Firewalls, proxies, application level gateways, and even intrusion detection systems are designed to deter, inhibit and/or detect an external adversary from gaining access to the local network. Once an attacker is able to gain access to the local network, internal defenses are limited in their efforts to prevent the misuse of resident network resources.

An adversary can use the protocol information that is displayed by querying end hosts, or by monitoring communications between hosts to extract information. This information can then be used to enhance his capability of attacking these hosts, and/or the services residing on them.

A new technology called Dynamic Defenses using Dynamic Network Address Translation (Dynat) is being heralded as a means of protecting networks from attacks by an external adversary. This is accomplished by moving away from protecting stationary addressable end hosts, to providing a means to continually change the end host address and other protocol-associated fields. This periodic changing of the communication protocol fields is intended to prevent an adversary from attacking any individual host because only authorized users have the proper information that correlates proper protocol field mappings.

There are a variety of ways to implement this technology. This report identifies these approaches by providing a description of important attributes that encompass the Dynat technology. These identified attributes provide a means by which an analysis of this technology can be made. With an understanding of the important attributes, a classification tree can be generated. This tree allows the individual to identify the appropriate “Dynat type” under investigation. Once identification has been made, a description of its proper use, along with identified limitations and vulnerabilities can be discussed.

2. Dynat Process Control

The Dynat process control section describes how the Dynat control mechanism is implemented. Put another way, it is the technique that drives the Dynat process. The Dynat protocol is based upon the principle of changing network address and protocol fields over time (see section “*Field*

Obfuscation” for a list of these potential parameters). These parameters, such as the IP address, map to end nodes in the network. The end nodes represent Hosts or network devices located on the network. Nodes participating in the Dynat process have a means of synchronizing the changes to these network delivery parameters and are able to fully communicate with each other. The periodic changing of delivery parameters causes obfuscation over time. Thus, potential adversaries snooping a network (trying to gain information about hosts or services running on the network) are met with a barrage of dynamic parameters that are only understandable to participating nodes. The process control implementations of Dynat can be assigned one of three process control types: time base (synchronous), time base (polling) or packet/frame based.

2.1 Time Based Synchronous

Time based synchronous control approaches use a clock to determine when a Dynat encoding change is needed. The time element is referenced to provide timing for the rate of change of the Dynat code. Also, most importantly, its synchronization across all participating nodes is critical because the time is used to either index information that provides part of a dynamic encoding key or, in some implementations the time itself becomes part of the key. This key can be comprised of a single part that is the output of a time-based incremented index, or the time itself. The key can also be comprised of two or more parts. An example would be a static “secret” part that each participating node would share, and an additional part comprised of the time or an output of an index that is incremented by time. There could be even more inputs that comprise the overall key construct, but the dynamic part of the key would always change based on a common clock. Figure 1 illustrates the distributive synchronous approach to obfuscation control. See section *Key Types and Distribution* for a listing of the types of keys that can be used in a Dynat process.

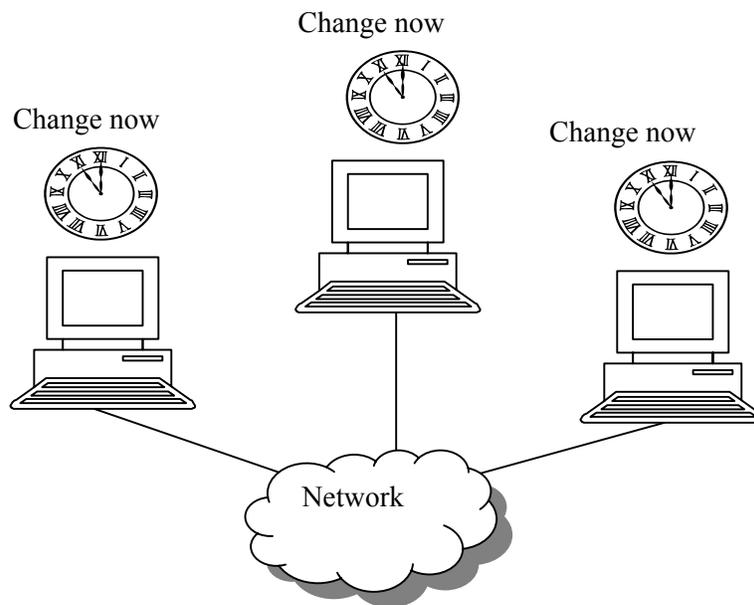


Figure 1

2.2 Time Based Polling

Time based polling control approaches also use a clock to determine when a Dynat encoding change is needed. This is implemented in a non-distributive fashion. It involves a controller node that is responsible for coordinating the changing Dynat values. The important distinction between a distributive timed sequence, as described in *the time based synchronous* approach, is that although the time element is referenced by the controller node to provide the timing for the *rate of change* of the Dynat code, the time does not play any role in the composition of the distributed key. The key is constructed independent of time. Time is used to control the obfuscation selection process and does play an important role in the protocol, but it becomes less critical than in the synchronous approach.

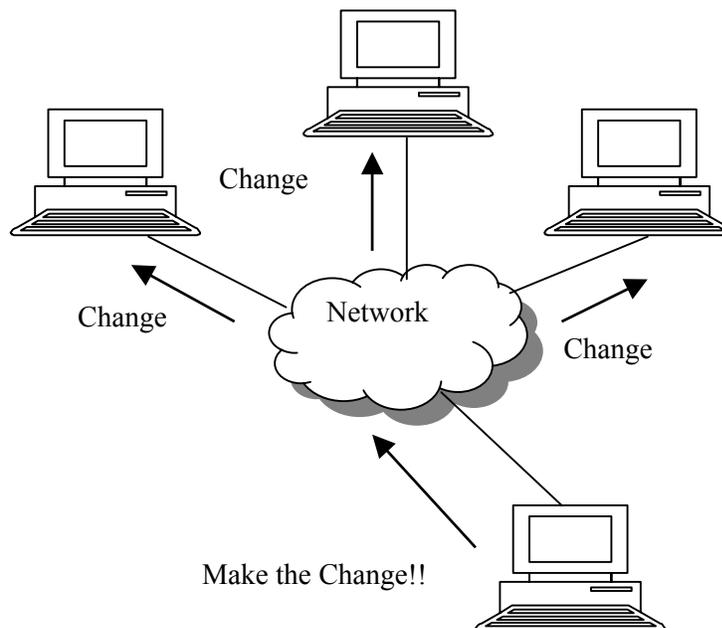


Figure 2

2.3 Packet or Frame Based

Packet or Frame approaches must involve a pre-determined action. This implementation has no need for any time or time-based coordination. The rate of change uses packets or frames as the finest granularity of change. This approach uses the number of packets or frames in each start up session to drive the Dynat change rate. And the packets or frames also provide the means to coordinate Indexing. Indexing can be utilized to step through variables that help construct the packet or frame key. The change rate is selectable but must be coordinated between each participating nodes. Some possible implementations including LAN and point-to-point are described in the following figures.

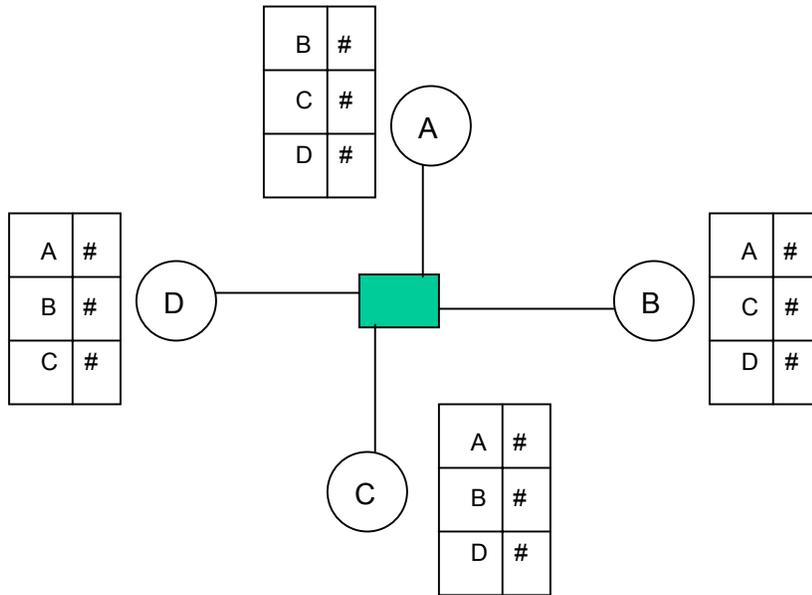


Figure 3

As seen in figure 3, each host has its own personal copy of a table that describes the number of packets that have been exchanged between itself and all other machines in the LAN. For example, machine B has information on the number of packets that have been transported from B to A, and A to B in one entry in the table. So every time B sends a packet to A, it increments its A value by one, and A also increments its B entry in its table by one as well. In this implementation, each machine has a piece of secret information that is only shared between each participating machine on the network, and it is through this piece of secret information that two machines are able to communicate with one another.

As seen in figure 4, this implementation works best as a connection-based implementation. The server keeps a table of the number of connections that have occurred for each machine. For example A's number in the table is the number of connections that have been made with machine A. If machine B wants to talk to machine C, B must request C's secret value that the server holds. When the server receives this request, it can assume that B is going to make a connection to C, so the server will increment C's value in the table and it will send that value to B. B will then try to make a connection with C. In order to communicate and read the message from B, C will need to request its connection value from the server as well. Once C has this value, B and C are able to communicate.

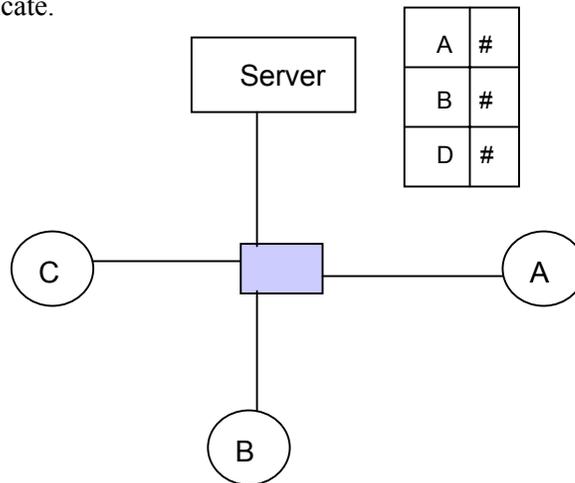


Figure 4

In the implementation shown in figure 5 each machine needs to be set in promiscuous mode and the network must be contention based. This way, every machine on the network is able to see all of the packets on the network. For every packet that each machine sees, each machine must increment their counter. This is then a secret value shared by all machines that they can use to implement the Dynat process.

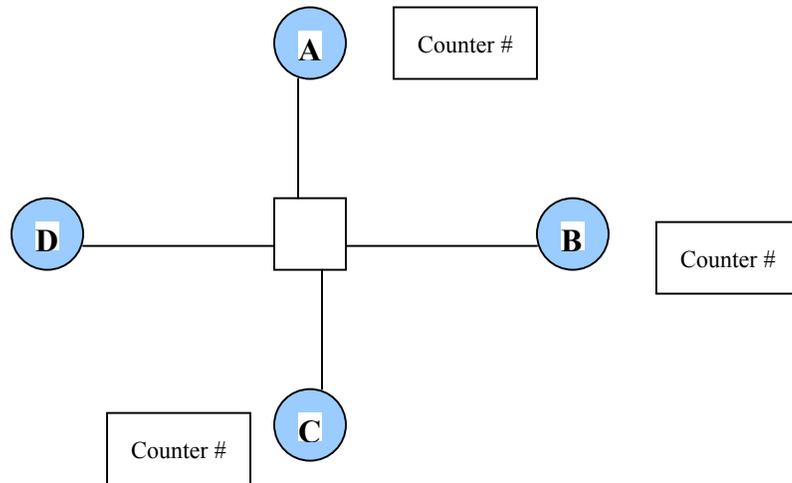


Figure 5

Another implementation, seen in figure 6, allows nodes in a point-to-point configuration two share a pre-arranged secret. How the address values are obtained and the order in which these address values are used is the secret. Packets are used as the counters that increment pointers to the table of values. The granularity of the address change rate can be implemented on a packet-by-packet value or a multiple of a packet. This approach can be used to connect Intranets or Extranets, VPN's, or any Enterprise backbone. An example of this approach is shown in Figure 6.

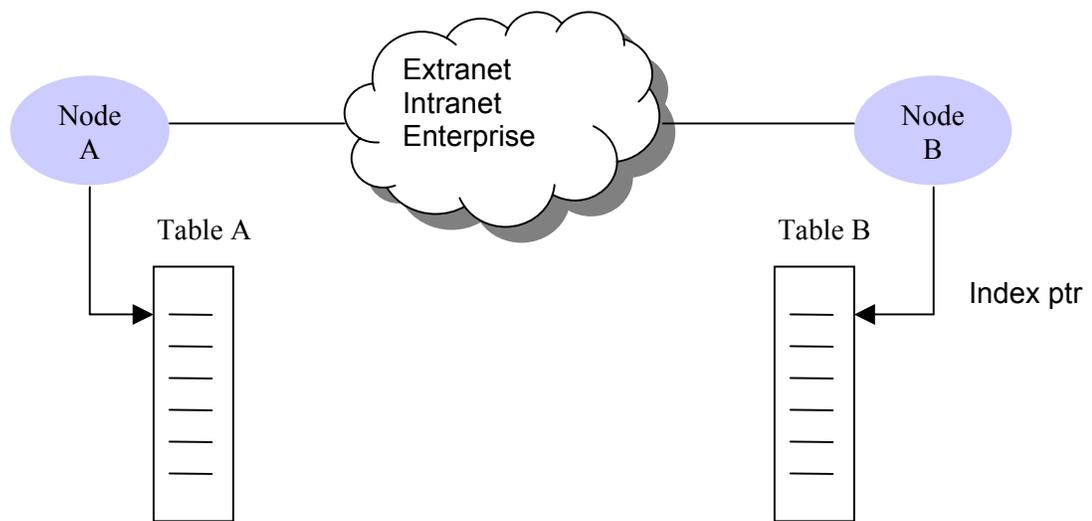


Figure 6

2.4 Key types and Distribution

When providing input to the obfuscation engine, there are different approaches to developing the “secret” or key and its distribution. These approaches are many but can be defined in the following ways:

- Local, single part, static (LSPS)
- Local, single part, dynamic (LSPD)
- Local, multiple part, static&dynamic (LMPS&D)
- Remote, single part, static (RSPS)
- Remote, single part, polling (RSPP)
- Combination, multiple part, static&dynamic (CMPS&D)

The LSPS key implementation allows for all participating Dynat nodes to be provided with an initial single part secret key that is administered locally and is not changed or updated by any automated remote control function. To change the key, a local approach is used. The LSPD key approach implements a local key that is dynamically changed based on a pre-defined time sequence. This time sequence can solely be used to reference the change, or the time itself could be used as the key. If time becomes part of the actual key that drives the obfuscation it is referred to as a “hash” key. The LMPS&D key is administered locally. All nodes must share a “static” secret part of the key, and then must reference a clock source that will be used to provide the second part of the key.

The RSPS key is administrated by a remote controller function that downloads a “startup” session key when requested by a remote participating node or distributed by a command from the network administrator. A new static key can be distributed at any time, but must be distributed to all participating nodes to enable communications.

The RSPP key implementation is more unique than other approaches. The key is not a key at all but is comprised of a control signal that is sent to all participating nodes. The actual key is a table of network protocol fields that are used in the obfuscation. When this control signal is polled on a pre-selected time sequence the tables are recreated based on the on-board obfuscation algorithm.

A combination key (CMPS&D) uses a local component and a remote component that are “combined” to form the overall key. The local component of this key is normally static, but can be implemented in a dynamic fashion. The remote component of the key is sent by a control function that distributes this part of the key to all participating nodes. Both of these parts are combined on the participating nodes to drive the obfuscation engine

3. Dynat Code type

The Dynat implementation type can be described as solely software based, hardware based or a combination of both hardware and software (hybrid). Although no “hardware only” implementations have been identified.

3.1 Software Approach

In the software implementation, all controller and obfuscation activity, which includes the obfuscation engine, is implemented in software. This software sometimes referred to, as the

“Dynat process” can be implemented on all participating workstations nodes, routers or gateway interfaces.

The software is written to create programs that capture outgoing packet streams and extract selected protocol field information. This information is then shifted through a obfuscation process that uses a key for “obfuscating” the original protocol field information. The “changed” field information is then inserted into the packet and/or frame from which it was originally retrieved. After a recalculation of the appropriate CRC information, the packet and/or frame is then sent out on the network. There is also a software process running on participating Dynat nodes that is responsible for the recovery and reconstruction of the original packet or frame. The reconstruction code is just a reverse of the obfuscation process, which is dependent on the type of process control being implemented. An example implementation of an algorithm used in a software approach is shown below.

Obfuscation parameter technique that performs initial conversion: ⁷

$$p^I_{NEW} = E_K(p^I_{OLD}, f)$$

where

– p^I denotes a concatenation of all p in P ,

– p^I_{OLD} denotes original parameter list

– $E_K(.)$ denotes encryption using session key K and

– f denotes obfuscating factor (e.g. pseudo-random number, time)

Obfuscation parameter technique that reverses initial conversion:

$$-p^I_{NEW} = D_K(p^I_{OLD}, f), \text{ where } D_K(.) \text{ denotes decryption using session key } K.$$

There could be multiple ways to implement the key construct from the above example as described in the section *Key Types and Distributions*.

3.2 Hardware Approach

A hardware only solution would use a hardware based obfuscation engine to process communication protocol fields that were of a pre-determined fixed size. The keying element responsible for the orchestration of the obfuscation would be imbedded into the hardware. Changing keys in this scenario would entail changing a chip or reprogramming a storage device. The hardware can be locally resident on the workstation, server, router or gateway or in a standalone device.

⁷ DSO National Laboratories

3.3 Hybrid Approach

The hybrid approach uses a hardware based obfuscation engine driven by a software based controller interface. The hardware can be locally resident on the workstation, server, router or gateway or in a standalone device. The hardware implements the obfuscation algorithm for the participating fields of the communication protocol stack. The software-based interface is responsible for the delivery of the keying elements. The type of keying protocol selected may vary. See section *Key Types and Distributions* for a list of possible choices.

4. Architecture Support

The identified architectures that the Dynat technology exist in include the following:

Architecture Type	Level
LAN segment switched	access
LAN segment hub (contention based)	access
LAN- to- LAN (Routed/layer3/4 switched)	distribution
Gateway- to -Gateway	distribution/core
Segment-to-Segment	distribution

The actual implementations within the above listed topologies vary.

4.1 LAN Segment Distant Server

Some implementations include a switched LAN segment that contains all client machines running a “Dynat process” with no servers on the same segment. All the local servers that support the services reside on another LAN segment. This directs all client/server interactions across a routed interface where the other associated Dynat process resides. This routed interface can be connected to another local LAN segment as seen in figure 7, or the routed interface can be connected to the Internet for a remote connection to a distant server as seen in figure 7.a.

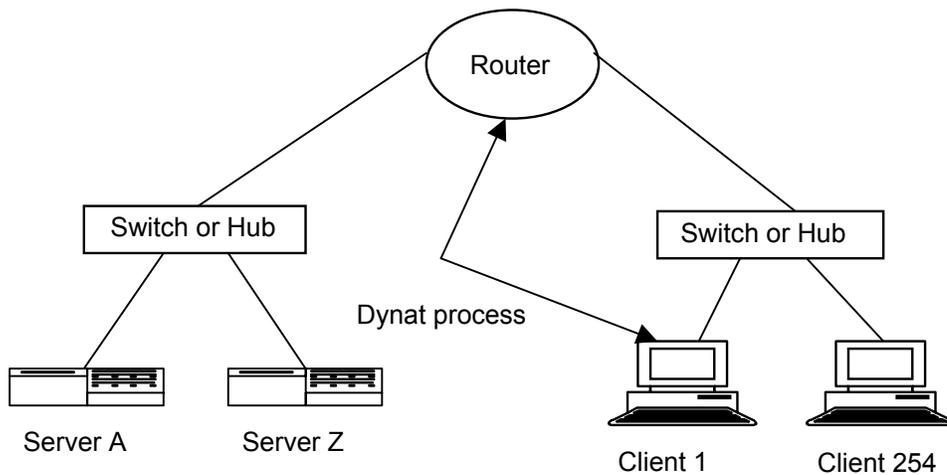


Figure 7 LAN Segment Distant Servers LSDS

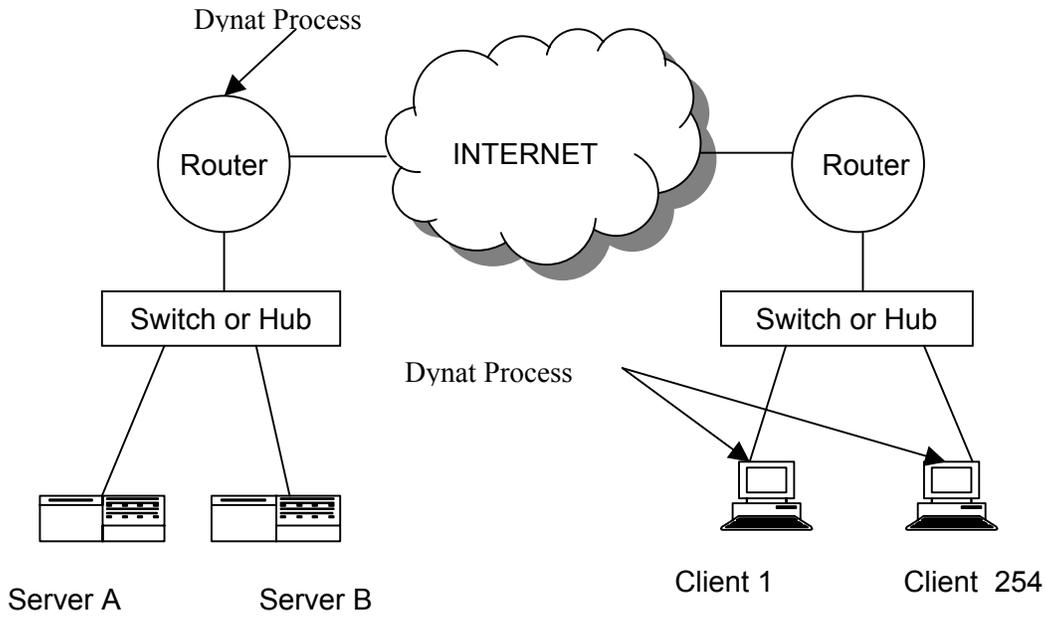


Figure 7.A LAN Segment to Distant Servers LSDS

4.2 LAN Segment Local Server

Another implementation uses a more traditional approach to the client/server architecture by allowing both clients and servers to reside on the same local segment. There is no need for a router, and the network can be expanded by the addition of Switches or Hubs. Each participating node in this configuration is running a Dynat process. Communications to each node is provided through the Ethernet Switches or the contention based Hubs. This topology provides the least amount of complexity for the network administrator. Figure 8 depicts this approach.

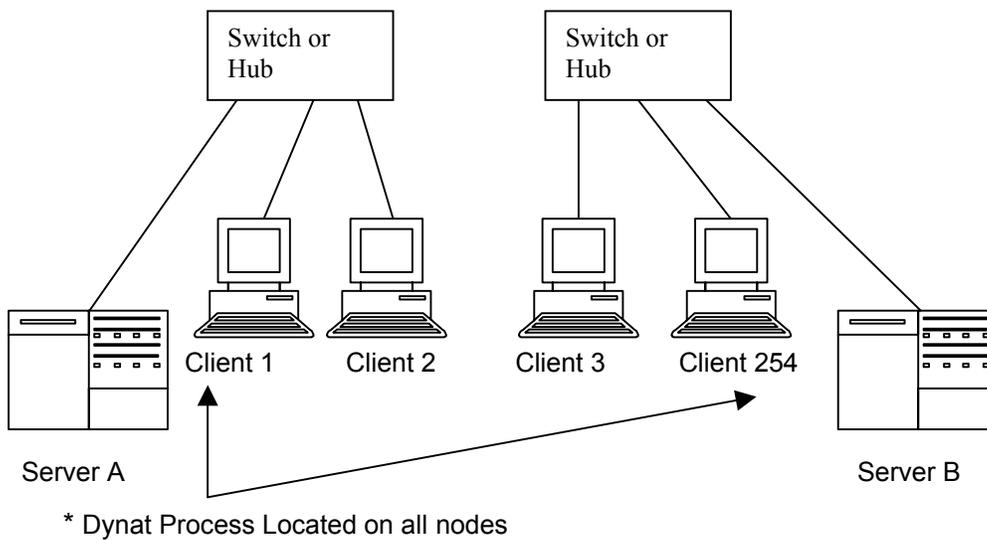


Figure 8 LAN Segment Local Server LSLS

4.3 Local Router to Local Router

Another similar approach installs the Dynat technologies on two directly connected routers. Again client and servers are located on different segments but neither are hosting the “Dynat process”. The Dynat process is active between the routers interconnecting link. This technique is utilized to obfuscate a link that maybe exposed to an outside or unprotected enterprise or campus network environment. The LAN segments are considered isolated from the exposed environment. Figure 9 displays this implementation.

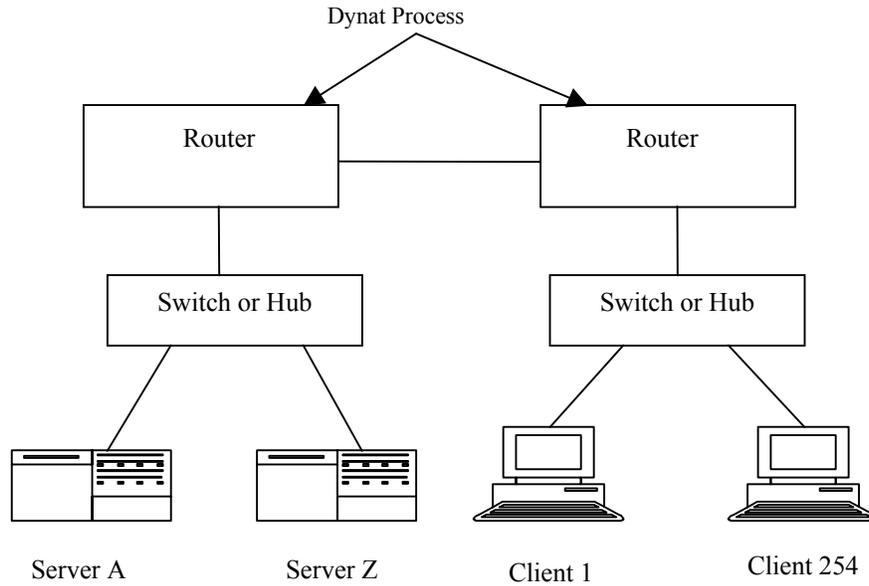


Figure 9 Local Router to Local Router LRLR

4.4 Gateway to Gateway

The term “Gateway” infers that this interconnection between routers is not between local resident LAN segments. For the most part this configuration is presented as external networks connected over a public medium, i.e. Internet. This connection could exist between two common identities and would be referred to as an Intranet connection. It could also represent two or more different identities and be referred to as an extranet connection. Regardless of the end identities, the Dynat protocol is implemented on the interconnected gateways and provides a means of obfuscating end node address assignments across a public medium. This configuration is shown in figure 10.

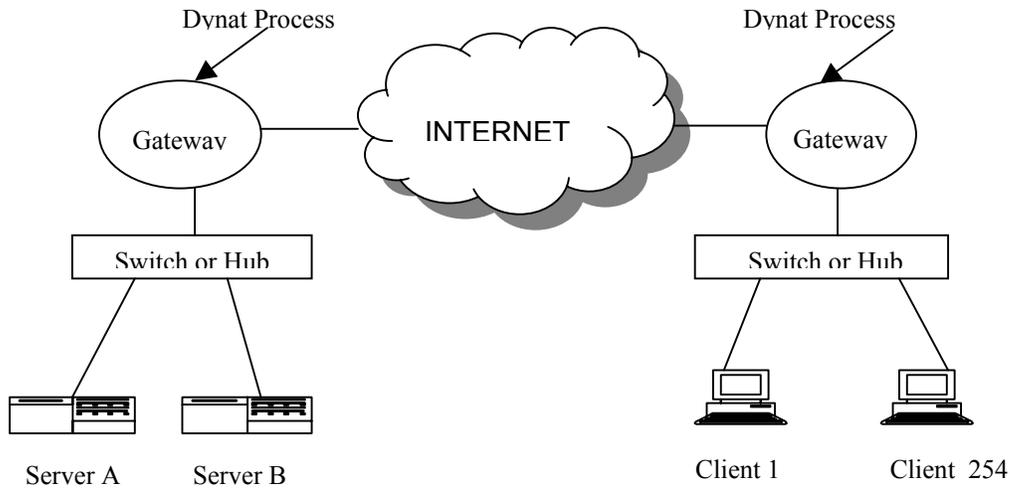


Figure 10 Gateway-to-Gateway GG

4.5 LAN segment to LAN segment

One slight variation to all previously described Dynat topologies involves two or more distinct segments implementing the Dynat protocol on all hosts and maintaining this obfuscation across interconnected Routers or Gateways. This infers that the distinct Dynat processes must provide coordination between their interconnected segments. Routers and gateways must also participate in the Dynat protocol as well. This participation maybe located on each Router or Gateway or in the form of an external “stand alone” interface as shown in figure 11 below.

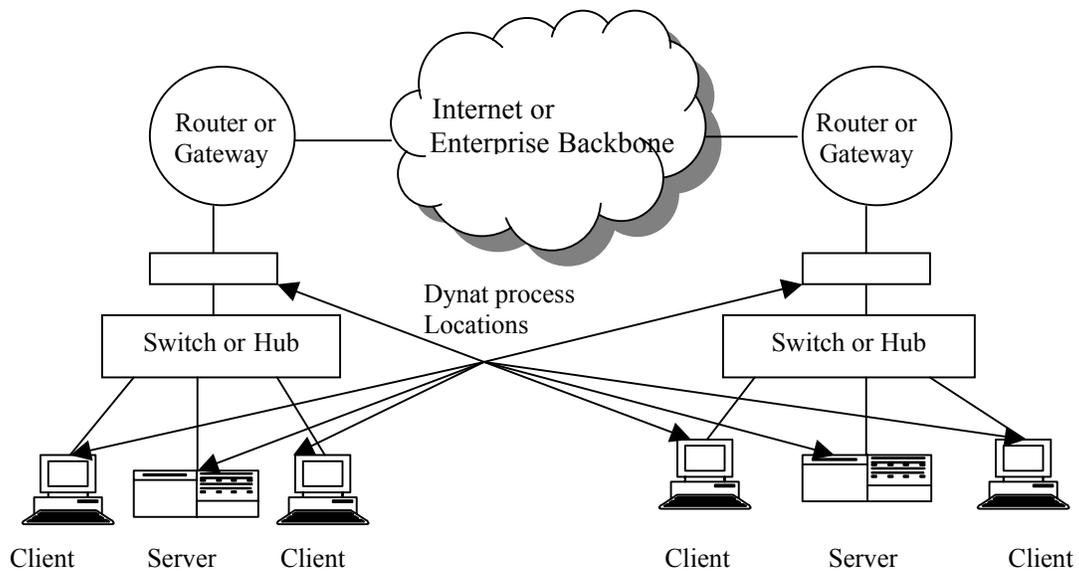


Figure 11 Segment- to- Segment SS

5. Field Obfuscation

There are many fields within the standard networking protocol stack that provide input to the obfuscation engine.

5.1 Protocol Fields

The following is a list of protocol fields that have been identified for potential use in the obfuscation process.

Layer 2	MAC address	source	destination
Layer 3	IP address IP Type of Service field	source	destination
Layer 4	TCP TCP sequence number TCP window size UDP	source port source port	destination port destination port

It is important to note that all identified implementations of Dynat use multiple fields to provide the obfuscation, although not all fields cross network layer boundaries.

6. Performance Degradation

No security technology that is added to aid the protection of a network, process, or device is inserted without some measurable form of hindrance or degradation. This hindrance can manifest itself in multiple ways. In the Dynat protocol this hindrance or degradation can be shown in the network protocol processes, in the applications that are supported across the network, and the security devices found in modern networks. The amount of degradation is dependent on the type of Dynat implementation. As the techniques that are being used vary, so will the difference in the interaction between applications and underlying network protocols.

6.1 Network Protocol

Some Dynat protocols may have an adverse affect on network performance. This network degradation is not based on processing delays seen with software based Dynat implementations, but is noticed within a specific layer of the network protocol stack. This degradation is found when the obfuscation process includes the Ethernet Media Access Layer (MAC) of the network.

This layer is adversely affected due to the interaction of this protocol layer with network-associated equipment, specifically Ethernet switches. When Ethernet switching is used to provide network communications, the switch builds a content addressable memory (CAM) table that contains a mapping of the source MAC address of an Ethernet frame and its associated port. This is needed to allow the switch to determine the destination port of a transmitted Ethernet frame. All workstations and servers on a local segment have a unique MAC address, which is associated by the Ethernet switch to its interconnected port. This mapping allows the switch to direct the Ethernet frames to their proper destination. The information to build the CAM tables is retrieved

by the switch when the attached Ethernet host needs to resolve a destination IP address with its destination Ethernet address. This is done by a process called Address Resolution Protocol (ARP). When an ARP frame is transmitted from a host, it contains the following structure as seen in figure 12.

Ethernet Dest Addr	Ethernet Src Addr	Sender Ethernet Addr	Sender IP Addr	Target Ethernet Addr	Target IP Addr
Field 1	Field 2	Field 3	Field 4	Field 5	Field 6

Figure 12

The first field in the ARP frame contains the broadcast address of all ones. This alerts all participating hosts in the network that an address resolution has been requested. This also tells the Ethernet switch to present this frame to all ports so it can be seen by all hosts. The second field contains the MAC address of the hosts requesting the resolution. Field 3 contains the sender's Ethernet address, which in most cases is the same as field 2. Field 4 contains the sender's IP address, and field 5 is empty and needs to be filled with the MAC destination address that is addressed to the target's IP address found in field 6. Since this information is presented to all hosts the hosts can build an ARP cache containing MAC-to-IP address mappings by simply recording field 3 and field 4. The Ethernet switch also gains valuable information that is needed to populate its CAM tables. The information recorded by the switch is the mapping from the Ethernet source address contained in field 3 with the port of origination of this frame, along with the mapping of the target Ethernet address to the port response contained in field 5. This allows the switch to deliver any additional frames with these two-recorded addresses to the right destination ports. If the workstations that contained the previously identified MAC addresses were physically swamped without the knowledge of the Ethernet switch, the transmitted information could possibly be sent to the wrong destination.

There are mechanisms built into the system to help mitigate these circumstances. First, each host that caches information allows for that information to be timed out. This helps prevent the stale address assignment to occur by forcing an ARP from the transmitting workstation. Second, an Ethernet switch that notices two identical MAC addresses assigned two different ports will only accept the latest entry as current.

Network performance degradation occurs when employing a Dynat protocol that uses the MAC address field for obfuscation. The degradation occurs when the Dynat protocol must force an ARP whenever the MAC address of a host is changed. This ARP is needed to allow the Ethernet switch to update its CAM table and for distant hosts to update their ARP caches. Without this forced ARP the Ethernet switch would be unable to accurately deliver Ethernet framed information. This "ARPing" produces a three-fold performance hit. First it forces every host to process the ARP, secondly it adds additional entries into the CAM tables of the Ethernet switches, and thirdly it uses valuable port and bandwidth resources. The overall quantity of these additional ARP's are directly proportional to the number of clients participating in the Dynat protocol and the rate of change that is instituted by the Dynat protocol. This has an overall performance impact on the network, associated with the need for additional ARPing that can be described by the following formula.

$$D = \text{Network degradation (Based on additional ARP's)}$$

$$\begin{aligned}\Delta T &= \text{Dynat change interval (seconds)} \\ N &= \text{Integer represents the number of Participating hosts} \\ D &\propto 1/\Delta T (N)\text{hosts}\end{aligned}$$

Another important observation refers to CAM table management. In some Ethernet switches filling up all the CAM table memory, which is caused by a rapid increase in the number of ARP transmissions can cause the switches to lock up. (See *MAC Attack* description in the *Adversarial Impact* section located in *Appendix C*.)

6.2 Application

Some of the issues concerning implementation of the Dynat technology revolve around how standard Network Address Translation (NAT) techniques have adversely affected applications in the past. Applications between distant communicating nodes, primarily applications that contain IP address and port information in the data payload, were adversely affected when “NATing” was implemented. This happened because the original IP address or port number was changed, but the information in the data payload that also referenced the original header remained unchanged. This caused an incompatibility problem which adversely affected application interaction. Some of these problems could be resolved with the use of application-level gateways that were able to transpose the information in the data payload, so the new IP header and port information was realigned correctly, but any application using encryption could not be fixed with the insertion of an application level gateway.

NATing affected other types of applications such as FTP, H323, Session Initiation Protocol (SIP) and Real Time Session Protocol (RTSP) because they all use a control port along with a data port to implement connections. When establishing a session these applications exchanged address and port control information before the establishment of a data session. This interdependency of these “bundled sessions” were not known by the NAT function and caused the sessions to be aborted. This occurred because the control function created data sessions that would originate in a direction that was not permitted by the NAT.

Other previously affected applications included those considered as peer-to-peer. When peer applications were distributed across private and public address space, the NATing function only recognized the establishment of a connection originating from the private realm. When a session was initiated from an external public location, as would be likely with a peering application, the NAT function was unable to resolve the end connection. Examples of peering applications include Instant-Messaging and IP telephony.

Internet Protocol Security IPsec has traditionally had problems transversing a NAT. This is because NAT’s change the IP source address field within the IP header and the IPsec Authentication Header AH is designed to detect changes to the IP header. This change is detected because IPsec-AH calculates a hash across the original packet (including the header) and stores this hash as part of the payload. When the NAT function changes the IP source address enroute to its destination this change is detected on the distant end and the packet is discarded. This also occurs when using IPsec in the Encapsulated Security Payload (ESP) mode. Because the original payload has a checksum calculated across the headers and data fields, this checksum could not be recalculated by a NAT function after an IP address has been changed because the original checksum has been encrypted. For more discussion concerning the Dynat technology and IPsec, see the section entitled *IPsec in the Security chapter* of this report.

The X-windowing system has also traditionally had a problem functioning over a NAT implanted link. The X-windows application consists of two distinct parts, the X server and one or more X clients.

It is important to note that the client and server operate in an opposite fashion from standard client server applications. The server controls the display directly, and is responsible for all input/output via the keyboard, mouse or display. The clients, on the other hand, do not access the screen directly, they communicate with the server, which handles all input and output. It is the clients which do the "real" computing work, running the graphical applications. The clients communicate with the server, causing the server to open one or more windows to handle input and output for that client.

The X server runs on the machine to which the monitor is connected. The clients may also run on this machine, communicating directly with the server. On most workstations, this is the normal situation. However, X is a networked window system, and it is possible for the client to run on a remote machine, communicating with the server via some form of network. In most cases, this would be via TCP over IP. To direct an X client to communicate with a distant server, a display variable must be passed to the machine hosting the application to be displayed. The X terminal transmits an IP address from the client to the server to set this display variable. The display variable is sent inline during a Telnet session. This display variable contains the IP address of the machine that will be hosting the X server. In normal use, a user will access the corporate LAN from outside the network or alternate place of work to access applications on the corporate local LAN. In this scenario, the X client is running on a host on the public side of the NAT and the X server is running on a host on the private side of the NAT. Since the display variable (which is the IP address of the remote host) is transmitted within the data portion of the IP packet, the NAT function is unaware of this IP address and when the windows server tries to acknowledge receipt of an X command from its distant client, the NAT will be unaware of this IP address and discard the packet. This becomes even more difficult to resolve when the initial session setup uses Secure Shell (SSH) instead of telnet.

When discussing incompatibility issues concerning the new Dynat implementation it is important to remember that above mentioned applications may encounter problems only when transmitting or receiving data from *outside a local segment*. For example, the LAN Segment Local Server (LSLS) topology previously discussed in the *Architecture Support* section of this report, would not be affected because all communications are constrained to a local LAN, thus preventing the potential packet rejects that could occur at the router or gateway interface.

It is important to remember that standard packet filtering techniques associated with Routers and Gateways can be adversely affected by a Dynat protocol. The location of the Dynat process along with coordinating route and filter functions is critical in maintaining a Dynat protocol across subnet boundaries. See the following *Security section* for proper Dynat insertion.

6.3 Security

It is important to note when integrating the Dynat technology into an existing network that some of the previously installed security devices used to protect specific network topologies may be incompatible with the Dynat protocol. The following is a list of the topologies and network security device configurations that will be affected when implementing Dynat.

6.3.1 Gateway to Gateway

If a Dynat device is used to obfuscate network information across two inter-connecting gateways, the Dynat protocol cannot be implemented prior to the gateway's VPN or Firewall function. This is because VPN gateway implementations require that each participating gateway have a static, authenticated termination at each other's node interface. And Firewalls can either provide a proxy function with respect to an application level firewall, or more rudimentary functions such as keeping state information about each originating connection. Both of these activities cannot be implemented when all the network information needed for session tracking is continuously being changed. Figure 13 shows the *improper* insertion of the Dynat technology, while figure 14 shows a functional implementation.

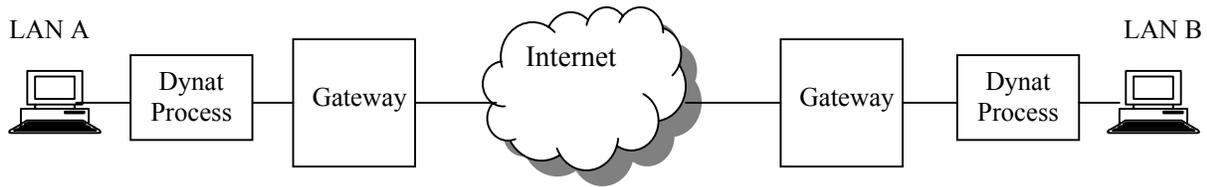


Figure 13

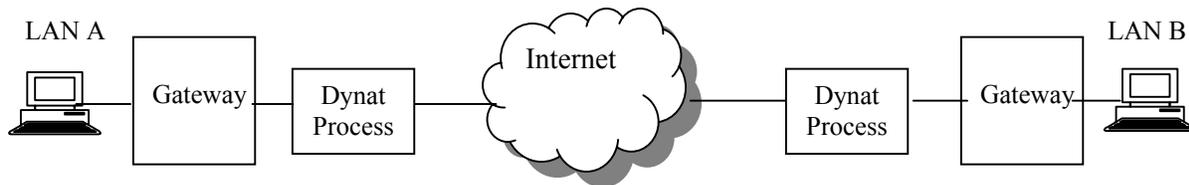


Figure 14

6.3.2 Local Router to Local Router

The Local Router to Local Router implementation would be very similar to the Gateway approach. The “Firewall” on the routers would be in the form of an access control list (ACL) that would provide the necessary filtering of traffic. Again, for this operation to be effective the Dynat process must be located on the interconnected link between the routers. Otherwise the routers could not perform ACL packet filtering functions or any type of proxy function like standard network address translations (NATing) . It is important to note that although the router cannot perform any type of packet filtering of static IP addresses, it can still maintain the ability to route packets. As long as the subnet mask, being used on the local segment to differentiate network ID and Host ID, is the same subnet mask applied at the router interface, the router will be able to route the network ID portion of the IP address. Figure 13 shows insertion of the Dynat technology across a routed link that does not affect a routers ability to perform packet-filtering activities. Figure 14 shows the insertion of the Dyant technology that inhibits the ability of a router to perform packet filtering through the configuration of Access Control Lists, but allows standard forwarding table look-ups for normal routing functions.

6.3.3 LAN Segment Distant Server

The LAN Segment Distant Server topology needs the intervention of a routing function to allow local LAN segment workstations to reach services located on a different segment. As mentioned previously in discussion of this topology in the *Architecture Support* section of this document, the Dynat process can be integrated with the routing function where the packet forwarding process of the router is not inhibited by the obfuscation. The Dynat process can also be located on a standalone device that precedes the routing function, or in a standalone device that follows the routing function. If the Dynat process precedes the routing function, the router will be unable to perform any type of packet filtering, but will still be able to route. If the Dynat process follows the routing function, the router will be able to participate in any host based filtering configured through a standard access control list, as described earlier in the LRLR topology configuration. If a network-based Intrusion Detection system is needed to perform traffic analysis on the segment hosting the Dynat process, it must participate in the Dynat function to be able to gain any usable information. See the following *IDS Integration* section for more details.

6.3.4 LAN Segment Local Server

The LAN Segment Local Server topology self-contains the Dynat process; it does not require the services of a router or gateway to perform its normal operations. Therefore any Firewall or Proxy server function will be unaffected by this implementation. Although it is important to note that if network intrusion detection services will be located on this segment, the intrusion detection system *must* participate in the Dynat function to gain any usable information available on this segment. See the following *IDS Integration* section for more details.

It is important to note that although obfuscating the MAC layer within a LAN segment increases the difficulty of end hosts and service mapping, it will also prevent a security technique associated with Ethernet switches from functioning. The security technique is called MAC port locking. This feature allows MAC addresses to be associated with specific ports and will only accept data from these specific ports when the source MAC address field matches a pre-defined filter. If the MAC address is continually being changed during the Dynat process, this feature *must be disabled* to allow proper data flow within the Ethernet switch.

6.3.5 Segment to Segment

The LAN Segment to LAN Segment combines the services of the local LAN segment, with reach capability to another LAN segment, while still maintaining the Dynat protocol across segments. This implementation has some of the characteristics of the LAN segment Local server and the Gateway-to-Gateway topologies. This implies the same local security concerns as previously discussed in the LAN Segment Local Server section also apply to this approach. When the obfuscation crosses segment boundaries the same Router or Gateway limitations must also be taken into account. Review the Router-to-Router and Gateway-to-Gateway topologies for a discussion of these concerns.

6.3.6 Label Based Switching

This section will discuss the techniques of packet/frame delivery that relies on a “label” assignment for each packet or frame and the impact of this method with the use of Dynat.

6.3.6.1 MPLS

Multiple Protocol Label Switching (MPLS) is a protocol that assigns labels to packets. The label is assigned at the access point to the network called the “ingress” and is referenced, and acted upon, as the packet is switched from node to node until it reaches its exit point called the “egress”.

This label technology can be implemented on a wide range of network sizes, from a few network subnets located on an Enterprise network, to dozens of subnets implemented on a large Internet Service Provider (ISP) network. These labels represent packet flows that are comprised of individual connections or an aggregation of connections.

Label-to-flow assignments are created before the flow of data (packets), by an underlying label assignment protocol. These labels are assigned and inserted between the Ethernet frame and IP packet and are referred to as a “shim” header. Because this shim label field is re-written or changed as the packet traverse from one node to another it cannot be hidden or its label value changed from one label-processing node to another.

Because labels can convey information about the packets they are switching such as quality of service, it can be used by an adversary to identify important flows. Because label assignments usually occur at the boundaries of networks (ingress) any Dynat process that was initiated at a host will not be able to participate in obfuscating the label process as seen in figure 15.

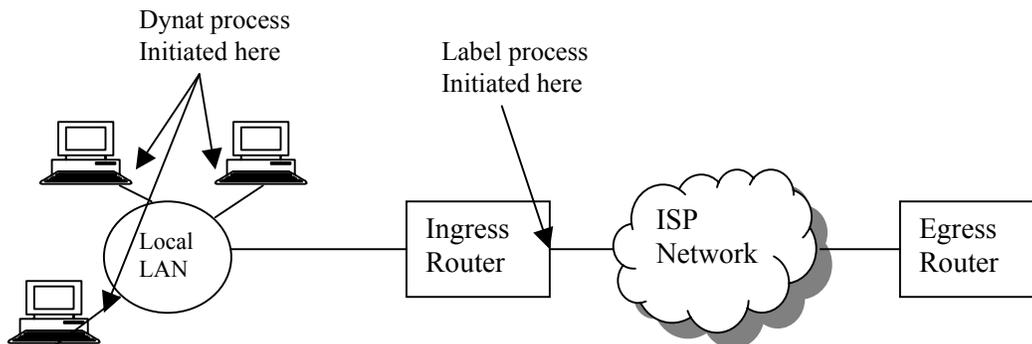


Figure 15

Although impractical, it may be possible to implement a Dynat process on a pair of inter-connecting label switch routers. Keeping in mind that the Dynat process providing the label obfuscation has to be implemented on a link-to-link basis. This maybe desired to provide protection against a particular link that is exposed to adversarial observation. Figure 16 illustrates this approach.

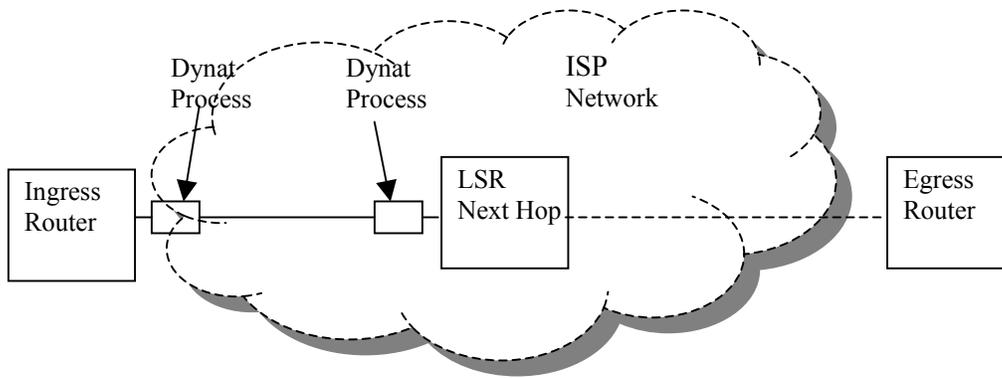


Figure 16

The Dynat process can be included as part of the label switch routers (LSR) or can be deployed as a standalone device between the participating LSR's. Regardless of the deployment strategy the Dynat process must be reversed prior to the next hop label processing.

6.3.7 VLAN

With the advent of layer two switching environments every device no longer sees all packets that are transmitted across the network. This is because collision domains are isolated at each individual port. Because of this isolation, the distant limitations that were implemented and enforced in older "contention based" networks (nodes within a common collision domain) are no longer in effect, allowing layer two switch networks to grow very large. As the sizes of the networks have grown so has the number of broadcast packets that must be processed by each end station. Broadcast domains are not isolated to individual ports but are present throughout a common sub-net. The management of individual devices becomes difficult, especially if they have different security requirements and the isolation of defective devices becomes more difficult as all devices are associated within the same broadcast domain. To address these problems Virtual Local Area Networks (VLAN) have evolved.

A VLAN can assign individual ports to VLAN groups within a common broadcast domain. This allows only members of the assigned groups to send, receive, and broadcast information to each other. This can also enforce security filters between different VLANs by forcing different VLAN packets to be processed at layer three where appropriate security filters can be installed.

6.3.7.1 802.1Q

The standard way of identifying VLANs is by inserting a VLAN identifier into the frame header referred to as frame tagging. The IEEE 802.1Q frame-tagging format provides a standard way of identifying frames that belong to an individual VLAN. This format allows for interoperability between multiple vendors. Because the 802.1Q technique uses an internal tagging mechanism, which modifies the existing Ethernet frame but does not perform encapsulation, it can be switched throughout any vendors Ethernet based network.

When a Dynat process is implemented at the MAC layer it obfuscates the source and destination addresses of the MAC frame. This technique helps in the prevention of endpoint identification and thus any service that maybe using this end point. But with the implementation of an 802.1Q VLAN frame an adversary can now have an additional means of identifying an Ethernet frame.

By capturing the Ethernet frame and reviewing the tag information contained within the frame header, the VLAN membership can be identified and may lead to additional means of user group and traffic identification. Because this tag is inserted at the switch port of the Ethernet switch and is acted upon by other switches in route to its final destination, it cannot be modified by the Dynat process.

6.3.7.2 ISL

The Inter-Switch Link (ISL) protocol is a Cisco proprietary method of identifying VLAN membership across a switched environment. The method for identification is by encapsulating the entire Ethernet frame with a 26-byte header. This header contains the VLAN information needed to deliver it to its proper destination port. Because this is a proprietary way of performing VLAN identification this method can only be used on Cisco supplied switches.

The impact to the Dynat protocol, which is manipulating the MAC layer for end point and service obfuscation, remains the same as with the 802.1Q implementation mentioned above. An adversary has an additional means of correlating an end point to a user group or service. This has the effect of neutering the MAC obfuscation provided by the Dynat process

Static VLANs

Another name for static VLANs is port based VLAN. Both the ISL and 802.1Q discussions were based on Static VLANs. Static VLANs are created by assigning a port to a VLAN. This assignment is implemented as the Ethernet frame enters the port. The impact with using a Dynat protocol with this type of VLAN has already been discussed

Dynamic VLANs

A Dynamic VLAN uses the concept of a server to assign members into a VLAN. Dynamic VLANs can use the source IP or MAC address of the requesting device to assign it into the appropriate VLAN group. This method allows for end host to move from one Ethernet port location to another and still remain in a predefined user group.

A Dynat process that is implementing a MAC layer obfuscation when the distant server is using source MAC address for proper VLAN assignment will be serverly impacted and may either assign the requesting end device into the improper VLAN or simply reject the device. The same holds true for VLAN assignment based on IP address. This has the effect of disrupting the security associated with this approach.

7. IPSec Environment

IPSec provides security services at IP layer using two protocols, namely IP Authentication Header (AH) and Encapsulating Security Payload (ESP). AH provides connectionless integrity, data origin authentication, and an optional anti-replay. ESP may provide confidentiality, limited traffic analysis, connectionless integrity, data origin authentication, and an optional anti-replay. Both AH and ESP provide means for access control, based on distribution of cryptographic keys and management of traffic flows relative to these security protocols. [KA98a, KA98b, KA98c] Combining IPSec and Dynat could possibly provide greater security by harnessing security services of each technology. For instance, it may be desirable to encrypt highly sensitive data for

transmitting over the network, while providing a reasonable degree of anonymity and traffic analysis resistance. It may also be desirable to protect IPSec gateway from DDoS attacks.

7.1 Combining Dynat and IPSec

In this section an attempt is made to provide an insight on how better security can be provided by layering Dynat with IPSec V4, and the potential pitfalls in doing so. This discussion does not assume the technicalities of any specific implementation instances of Dynat.

In IPSec, security services offered on the traffic are described by a security association (SA). Each SA provides security services using either AH, ESP or both. Depending on security policy, two or more SAs can be created to provide AH and ESP protection in a nested manner. Two modes of SA exist: transport and tunnel mode, differing in the scope of protection. Basic combinations of SA are described in [KA98a].

Dynat, similar to IPSec, operates in a host or a security gateway environment. A Dynat session describes a single connection where Dynat processes at both ends are synchronized to communicate. List of topologies associated with implementations of Dynat technology are described in the *Architecture Support* section of this document. Nesting of Dynat session may be necessary under a security policy to provide different scope of protection when one or both endpoints are different. One such example is “LAN segment to LAN segment” topology, where host-to-host Dynat sessions are provisioned within gateway-to-gateway Dynat session. Nesting of Dynat session across same endpoints offers no additional protection, except improving access control with multi-layer protection.

IPSec SA and Dynat session may be combined to realize a security policy unachievable with a single technology. Multiple levels of nesting of the two protocols are possible too. They may be combined in all combinations and topologies supported in each technology. Layering is a simplistic approach in combining security services offered by the two technologies without changing their logic.

One constraint to this approach is that each IPSec SA or Dynat session should be nested completely within another IPSec SA or Dynat session, so as to allow proper recovery and reconstruction. As such, the inner SA or session has to be established for outer SA or session to communicate.

IPSec and Dynat interaction can be discussed in one of two ways: “Dynat over IPSec” and “IPSec over Dynat”. “Dynat over IPSec” refers to an outer Dynat session nested over an inner IPSec SA. Vice versa for “IPSec over Dynat”. Although the definition focuses on the Dynat and IPSec layers, we do not preclude the existence of other Dynat or IPSec layers nested over or within the two layers.

1. Dynat over IPSec

Changing fields in a Dynat session may affect IPSec SA selection during outbound traffic processing. IPSec process looks up security policy database (SPD) using selectors. For outbound traffic, allowed selectors include source and destination IP addresses, transport layer protocol, source and destination ports. If these fields are changed in Dynat process, then IPSec SPD lookup may fail.

To overcome this, one may use range and wildcard for specification of selectors in a SPD entry to include all values that may be assigned for the Dynat session, or use multiple entries if that is not

possible. In the earlier case, one may choose to process packets with different values for the selector using a single SA (or SA bundle), or multiple SAs (or SA bundles). The latter case would result in multiple SAs (or SA bundles) at least the number of SPD entries.

2. IPSec over Dynat

Dynat process should be able to recognize IPSec protocols, select a Dynat session using next header field in IPSec header and obfuscate IPSec header fields.

The validity of Dynat and IPSec combinations depends on the security requirements specified in security policy.

7.2 Inter-operational Issues

This section seeks to provide a high-level understanding on the usefulness of employing Dynat in an IPSec environment. However no assumptions are made on the strength of security services afforded by either technology.

7.2.1 Similar Protection Services

There are some overlapping in protection services afforded in both IPSec and Dynat, namely access control, network scanning resistance, and anonymity and traffic analysis resistance.

1. Access Control

Both AH and ESP are vehicles for access control. Shared secret key is used for authentication/integrity and encryption services. Knowledge of the key is required for processes to communicate. By managing the distribution of cryptographic keys, using key management protocol like Internet Key Exchange (IKE) [HC98], admittance of packets through the IPSec process can be controlled.

Dynat attempts to achieve its goal of anti-hacker by providing access control. Dynat process changes delivery parameters in packet headers over time in a certain change sequence, and access is only allowed from synchronized peers with the knowledge of that change sequence. The change sequence is derived from a shared secret, in conjunction with other information. As such, access control is management based on the distribution of shared secret, a process similar to IPSec.

2. Hindering Network Scanning

Dynat could hinder network scanning from adversary. It is provided based on access control: adversary would be unable to perform network scanning if he does not know packet delivery parameters at that time. This feature can be provided at Dynat gateway or host.

IPSec can also resist network scanning by controlling access as described earlier. IPSec security gateway prevents scanning of internal hosts from adversary on external network, or external attackers. IPSec process at host can prevent service enumeration via network scanning by not passing packets with incorrect integrity check value (ICV) to higher layer protocols. It, however, cannot prevent host discovery by scanning port for automatic SA and key management protocol

like IKE.

3. Anonymity and Traffic Analysis Resistance

Anonymity could be achieved in Dynat by removing association between location information and identity of end process. As identity and location information may appear in transport layer and above protocols, sanitizing these fields would be necessary. In this manner, identity of the real end process could be hidden from network traffic observer. Further, traffic analysis could be hindered by hiding endpoint, flow and other traffic information in packet header. There is however limitation on the IP address values that can be assigned to remain routable.

IPSec could afford some degree of anonymity and traffic analysis resistance too. Security gateways employing IPSec ESP tunnel mode (ESP-TM) not only hide host IP addresses but also encrypt the entire IP packets. This provides anonymity among the hosts behind the security gateway and hinders traffic analysis from external attackers observing the encrypted tunnel. IPSec also allows user to control the granularity at which a security service is offered. For example, a single encrypted tunnel can be created to carry all traffic between two security gateways, or a separate encrypted tunnel for each TCP connection between each pair of hosts communicating across these gateways. The earlier case provides better protection against traffic analysis in hiding traffic flows in aggregation.

Though, IPSec could offer little protection against attacker observing on internal network, or internal attacker. No anonymity is afforded, as IP address is cannot be hidden at hosts. Encrypting transport layer and above protocols offers limited traffic resistance, as traffic flows among hosts are not hidden (although different TCP connections between two hosts can use and be hidden within a single encrypted tunnel). Moreover, there are no anonymity and traffic analysis resistance for thesecurity gateway from external observers.

7.2.2 Benefits of Dynat in IPSec Environment

There are some advantages for employing Dynat in IPSec Environment.

1. Dynat improves anonymity and traffic analysis resistance

As mentioned earlier, IPSec offers little anonymity and traffic analysis resistance against internal attackers. Dynat could provide such protections to a certain degree by obfuscating packet header fields in LAN environment (i.e. IPSec over Dynat). In similar manner, Dynat could also provide the protection services to IPSec security gateway from external attackers.

2. Dynat affords protection against DDoS attacks that consume CPU cycles

The threat of DDoS on IPSec security gateway exists. IPSec AH and ESP(-TM) protocols requires derivation of ICV to authenticate data origin and integrity of the packet. Mandatory authentication algorithms, namely HMAC with MD5 and HMAC with SHA-1, are relatively computational intensive. The process has to be performed for every packet that arrives on its interface, especially when packets are sent in small size over high link speed.

IPSec ESP and ESP-TM poses no more vulnerable than AH to DDoS attacks. Although cryptographic algorithms would require much higher computation, IPSec process would authenticate ESP packets before passing them to its cryptographic module.

Fast reject algorithms may be designed in Dynat, as in the case of NetEraser, using knowledge of packet source to quickly reject DDoS packets without wasting resources. This is especially effective in frame or packet based Dynat systems, flooding could be more difficult as valid fields change on every packet, even if legitimate packets can be intercepted.

In contrast, time based (synchronous and polling) Dynat systems is only effective in scenarios where network sniffing is not possible (or not in the threat model), or the encoding change rate is high. Otherwise, timed-based Dynat systems may be vulnerable to flooding using packet header fields obtained via sniffing legitimate packets within the epoch period.

7.2.3 IPSec Strengthens Dynat

IPSec could strengthen protection afforded by Dynat.

1. IPSec improves anonymity and traffic analysis resistance of Dynat.

Dynat obfuscates location information in IP address fields. Apart from the address fields on IP header, identity and location information is also often found in higher layer protocols. For examples, user's name can be found in "From" MIME header, user's machine configuration in the "User-Agent" MIME headers, and user's email address as password for FTP transaction. Such information at IP packet payload are often sanitized, by removing explicitly using application-aware software logic, or encrypted. Data confidentiality is a pre-requisite if Dynat traffic analysis resistance is to have any significance.

IPSec ESP can be used for such purpose by encrypting protocols above IP layers, before sending them over the Dynat session (i.e. IPSec over Dynat). Similar protection can be afforded for the case of Dynat over IPSec, though some implementation issues have to be observed to avoid revealing flow information unnecessarily, which will be discussed in section 7.2.4.

2. IPSec strengthens hacking and network scanning resistance of Dynat and system

Dynat provides access control based on knowledge of obfuscated packet delivery parameters. Time based (synchronous and polling) Dynat systems obfuscate packet in a fixed manner until the next synchronization or polling control signal from controller node. There exists a period when obfuscated packet delivery parameters for each communicating pair remain the same for certain period of time. Likewise, packet or frame based Dynat systems also have a window when the packet delivery parameters are fixed, if its encoding change rate is selected to be multiples of a packet. Adversary observing network traffic would be able to modify packets in transit, replay captured packets, or generate malicious packets to scan, flood or intrude the hosts or gateways.

IPSec AH, ESP and ESP-TM can afford integrity and anti-replay protection to every packet. By applying IPSec to Dynat session with the above-mentioned configuration, in Dynat over IPSec, IPSec can strengthen Dynat's access control, thereby improving its protection against hacking and network scanning.

Also, IPSec would add as a second layer of protection in IPSec over Dynat setting, providing additional access control to systems protected by these classes of Dynat to counter the threat. However in this setting, encryption provided by IPSec ESP and ESP-TM would degrade Dynat's access control (see section 7.2.4).

7.2.4 Security Implications

There are some weaknesses and pitfalls when inter-operating Dynat and IPSec.

1. IPSec may reveal flow information

Dynat obfuscates traffic by changing packet delivery parameters. This may create the illusion to traffic observer that packets in a Dynat session belong to different flows (each flow normally has different packet header values), thereby hindering traffic analysis. However, in Dynat over IPSec where IPSec is applied to the Dynat session, flow correlation would be revealed if the packets were to send over a single SA.

For hosts, to maintain the traffic analysis resistance afforded by Dynat, one should choose to send packets from a Dynat session in multiple SAs. To achieve this, in IPSec inbound packet processing, the values in the inbound packet itself may be used in the SAs. This applies to AH, ESP and ESP-TM.

Using multiple SAs would improve traffic analysis resistance, but at the expense of manageability.

This would be particularly significant for security gateway managing large number of sessions. For IPSec tunnels between two security gateways, one may instead choose to send multiple Dynat sessions over the same SA. This would also achieve flow obfuscation, though in aggregation instead of splitting as in the host case.

If traffic analysis resistance is not part of the security policy, then one may send the packets from a Dynat session in a single SA. We also note that if both Dynat session and IPSec SA have the same endpoints, changing the fields that would be encrypted in the inner IPSec (ESP or ESP-TM) SA would not be helpful in resisting traffic analysis.

2. IPSec using automatic SA and key management reveals identity

In Dynat over IPSec setting, automatic SA and key management may reveal identity information of SA's endpoint, thereby compromising anonymity and traffic analysis resistance afforded by Dynat. Dynat would not afford IPSec process anonymity and traffic analysis resistance protection, as IPSec is in the lower layer in this setting. For instance, Internet Key Exchange (IKE) would have the legitimate IP address shown on its packets during its SA negotiation phases. IPSec would also reveal its identity information if aggressive mode were used in the phase 1 exchange [HC98].

3. IPSec degrades Dynat's anti-DdoS

As mentioned in section 7.2.2, quick reject algorithms uses knowledge of packet source to quickly reject DDoS packets. That is, destination Dynat process would drop non-conforming packet delivery parameters, without incurring further processing.

But if these fields were encrypted prior to field obfuscation, in IPsec (ESP or ESP-TM) over Dynat setting, such knowledge would be hidden, thus making it difficult to implement a fast reject algorithm. Taking an example with Dynat alone, TCP port *port_a* will be obfuscated to *port_a1*. Destination Dynat process may do the same computation, dropping packets with IP address other than *port_a1*. If the field is encrypted with IPsec ESP formerly, destination Dynat process may have to encrypt *port_a* and obfuscate the result to identify correct parameter for packet admittance decision. Moreover, encrypting the fields requires knowledge of algorithm and keying information. Such complex process would be hardly fast enough.

Dynat anti-DDoS mechanism has to rely only on the unencrypted fields, so that would weaken the protection afforded.

4. IPsec degrades Dynat's access control

Similar to anti-DDoS case mentioned above, Dynat cannot effectively provide access control on encrypted fields in an IPsec over Dynat setting. As such, ESP and ESP-TM reduce the number of fields available for access control, and thus weaken its protection. Nevertheless, IPsec as the higher layer can provide integrity of the packets. As such, in this scenario the degradation in Dynat's access control would not adversely affect the protection against hacking, virus and network scanning.

7.2.5 Summary

The pros and cons of inter-operating Dynat in IPsec environment are summarized in Table 1 and Table2 respectively.

Table 1 Pros in Dynat-IPSec Combinations

Strengths	Dynat over IPSec			IPSec over Dynat		
	AH	ESP	ESP-TM	AH	ESP	ESP-TM
Dynat affords anonymity and traffic analysis resistance				✓	✓	✓
Dynat affords anti-DDoS				✓	✓	✓
IPSec improves anonymity and traffic analysis resistance of Dynat by hiding packet payload and some header fields		✓	✓		✓	✓
IPSec strengthens hacking and network scanning resistance of Dynat and system	✓	✓	✓	✓	✓	✓

Table 2 Cons in Dynat-IPSec Combinations

Weaknesses	Dynat over IPSec			IPSec over Dynat		
	AH	ESP	ESP-TM	AH	ESP	ESP-TM
IPSec may reveal flow information	✓	✓	✓			
IPSec using automatic SA and key management reveals identity	✓	✓	✓			
IPSec degrades Dynat's anti-DDoS					✓	✓
IPSec degrades Dynat's access control					✓	✓

1. Dynat over IPSec

In this setting, with the inclusion of IPSec within the Dynat session, IPSec can strengthen hacking and network scanning resistance of Dynat and system. However, anonymity and traffic analysis resistance afforded by Dynat would be negatively impacted. In particular, IPSec automatic SA and key management would reveal identity information, thus compromising anonymity and allowing traffic analysis.

The validity of this setting depends, nevertheless, on the security requirement specified in security policy. One viable scenario is to establish a Dynat session from host to host at different LAN segments, and to encrypt the traffic over public network via IPSec ESP tunnel. One benefit of this scenario is that, with sufficient obfuscation, the remote peer could be hidden from internal network observer. The secure tunnel that traffic between the hosts takes would also be unclear to the observer, assuming that there are multiple peering security gateways.

2. IPSec over Dynat

Dynat may provide anonymity and traffic analysis resistance to nodes, both hosts and gateways, complementing protection services afforded in an IPSec environment. IPSec ESP and ESP-TM work well with Dynat in enhancing Dynat's anonymity and traffic analysis resistance. Although encrypting the packets may reduce Dynat's ability to perform access control, integrity afforded by IPSec at higher layer would guard against the shortfall. This is applicable whether IPSec or Dynat processes reside at hosts or gateways.

Deploying Dynat at security gateway in this setting may provide additional protection against DDoS attacks from external network. This protection service may be affected however, if ESP or ESP-TM were employed in the IPSec SA.

This section gave a high level understanding on the security of employing Dynat in IPSec environment. Although there are duplicating service services offered by both technologies, some synergy exists. Dynat does have some merits in providing anonymity, hindering traffic analysis and anti-DDoS. The use of IPSec may also enhance some Dynat's security services. We also discussed some interoperability issues that may weaken the duo.

The validity of Dynat and IPSec combinations depends on the security requirements specified in security policy. We have not covered the specifics of individual Dynat implementations. Their applicability requires additional exploration.

8. IDS Integration

Intrusion Detection Systems (IDS) can be divided into two different types: host based systems or network-based systems. Host based systems reside on the hosts and monitor user access and file or operating system functions. They are designed to detect anomalies during user operations such as improper file manipulations, kernel calls, or application access violations.

Network based systems normally reside on standalone computer systems and monitor all information that is sent across the network. They perform protocol decode and determine the types of protocols being used across the network along with the sequence of connection events. They have an on-board signature library that is used to help identify documented attack signatures. Adversaries attempting to identifying operating systems, applications, or services on the network often use stealthy approaches in attempting to identify these assets. It is often very difficult to determine if a network is being mapped for its host ID's and services. Adversaries can hide in the clutter of normal background traffic while they perform their mapping routines.

The general approach that adversaries use to attack information systems is well documented and understood. The first step requires the identification of active hosts on a network, followed by the available applications and services. Once the services have been identified the adversary proceeds to identify the Operating System (OS) and its current revision. Tools such as NMAP and Nessus are used to identify the Host by techniques such as the TCP SYN scan. This networking scanning technique is used to scan a local network segment by using a bare TCP ACK scan to locate all live hosts on a network. The TCP/IP protocol dictates that all TCP acknowledgements that are sent to an end host that does not have an active session with the sender are require to respond with a TCP reset (RST). This reset response is then recorded by the NMAP process as an active end node. The second phase of the attack is implemented with the purpose of identifying all active ports on the identified active hosts. This is done by simply initiating a connection request using the TCP synchronization (SYN) request to the desired application port. A SYN/ACK response indicates that there is an available service at the port. This is also recorded by the attack tool. The final phase, if needed, is initiated to gather information such as the type and version number of the operating system. This is done by improperly manipulating the bits in the TCP header used for session startup, progression, and termination. Once the system is identified, more aggressive techniques for its compromise can be initiated.

Intrusion Detection systems collect information from a variety of system and network sources, then analyze the information for signs of intrusion (attacks coming from outside the organization) and misuse (attacks originating inside the organization.)

Before the creation and use of the Dynat protocol it has been difficult to detect an adversary's network or port scan, especially when this scan is conducted in a stealthy manner, the more active the network the more difficult the task. This is difficult because a scan makes use of the same protocols used to allow communications between all end nodes on the network and its activities are difficult to extract and respond to in real or near real time way.

Some of the means that Intrusion Detection systems use to detect anomalous intrusions are by observing significant deviations from normal behavior. Host based and Network based Intrusion detection systems perform a variety of tasks that are implemented to detect inappropriate activities:

- Monitoring and analysis of user and system activity
- Auditing of system configurations and vulnerabilities
- Assessing the integrity of critical system and data files
- Recognition of activity patterns reflecting known attacks
- Statistical analysis for abnormal activity patterns
- Operating system audit-trail management
- Tracing user activity on the network
- Recognizing and reporting alterations to data files
- Identifying specific types of attack and alerting staff for defensive responses

Implementing the Dynat protocol on a network can have a two-fold impact on detecting and thwarting adversarial intrusions. First, the adversary's scanning tools required to implement network and port scans may be severely hampered in the mapping of IP addresses to hosts because these tools rely on an accurate host mapping before conducting additional port and operating system mappings. The mapping tool may be unable to correlate the recorded information because the tool will be confused because visible traffic that is cataloged would no longer be valid when the port and OS activity is launched.

If an Intrusion Detection system is made aware of the Dynat protocol, it will be able to detect anomalous activity. Extensions implemented within Dynat can allow for state information to be kept for packets that are detected operating outside the current active address realm. This state information can be used to create filters that filter out packets outside the current address realm. This could be taken further in an active manner with responses to originating packets, which could lead the adversary into a honey pot environment that could be used to covertly gather information about the intruder.

References

- [1] CERT Incident Note IN-99-07 January 15, 2001
- [2] Information Assurance, IA Experiments 0106 Document, The Impact of Dynamic Reconfiguration on the Effectiveness of Intrusion Detection Systems
- [3] Major June Experiment Document, Dynamic Defense, Dynamic Network Reconfiguration, May 11 1999.
- [4] Invicta Network Inc., *InvisiLAN Technology: LAN Protection System. Technology Summary.* <http://www.invictanetworks.com/invisilan.html>
- [5] RFC 3027 Protocol Complications with the IP Network Address Translator, January 2001
- [6] DSO National Laboratories, CHUA, Kuan Seah, Dynamic Defense Mechanism Grasshopper, June 6, 2002
- [7] DSO National Laboratories, Wong Yip Heng, Study on Dynamic NAT Systems, June 6, 2002
- [8] The Anonymizer. <http://anonymizer.com>
- [9] O.Berthold, H.Federrath, M.Kohntopp, *Project "Anonymity and Unobservability in the Internet"*, Workshop on Freedom and Privacy by Design CFP2000.
- [10] David Chaum, *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms*, Communications of the ACM, Volume 24 Number 2, February 1981
- [11] Y.Guan, X.Fu, R.Bettati, W.Zhao, *Efficient Traffic Camouflaging in Mission-Critical QoS-guaranteed Networks*. Texas A&M University. http://www.itoc.usma.edu/marin/Wshop/Abstracts/TP2_1.pdf
- [12] Dorene Kewley, Russ Fink, John Lowry, Mike Dean, *Dynamic Approaches to Thwart Adversary Intelligence Gathering*, DISCEX II, http://www.iaands.org/discecx_II/Briefs/13June/I&E/I&E_4_Kewley_DISCEXII_DYNAT.ppt, June 2001.
- [13] Vernon Loeb, *Web Security, Privacy Are Goals of CIA Effort*, Washington Post <http://www.washingtonpost.com/wp-srv/WPcap/2000-02/16/030r-021600-idx.html>, Feb 2000.
- [14] R.E.Newman-Wolfe, B.R.Venkatraman, *High Level Prevention of Traffic Analysis*. Seventh Annual Computer Security and Applications Conference, San Antonio, Texas, Dec 1991.
- [15] M.K.Reiter, A.D.Rubin, *Crowds: Anonymity for Web Transactions*. ACM Transactions on Information and System Security, 1998.
- [16] A.Pfitzmann, M.Waidner, *Networks without User Observability*. Computer and Security, 1987.
- [17] J.F.Raymond, *Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems*. Designing Privacy Enhancing Technologies: Proceedings of International Workshop on Design

Issues in Anonymity and Unobservability, 2001.

[18] M.G.Reed, P.F.Syverson,D.M.Goldschlag. *Anonymous Connections and Onion Routing*. In Proceedings of the IEEE Symposium on Security and Privacy, May 1997.

[19] B.Schneier, *Applied Cryptography*, pg 219, 1996.

[20] SecuriTeam, *SafeWeb Vulnerability, Fingerprinting Websites Using Traffic Analysis*, May 2002. <http://www.securiteam.com/securityreviews/5MP0D0K75Y.html>

[21] Craig Smith, Peter Grundl, *Know Your Enemy :Passive Fingerprinting*, Subterrain Siphon Project, <http://www.honeynet.org/papers/finger/>, March, 2002.

[22] P.F.Syverson, M.G.REED,D.M.Goldschlag. *Onion Routing Access Configurations*. DISCEX 2000: Proceedings of the DARPA Information Survivability Conference and Exposition, 2000

[HC98] D.Harkins, D.Carrel, *The Internet Key Exchange (IKE)*, IETF RFC 2409, Nov 1998.

[KA98a] S.Kent, R.Atkinson, *Security Architecture for the Internet Protocol*. IETF RFC 2401, Nov 1998.

[KA98b] S.Kent, R.Atkinson, *IP Authentication Header*. IETF RFC 2402, Nov 1998.

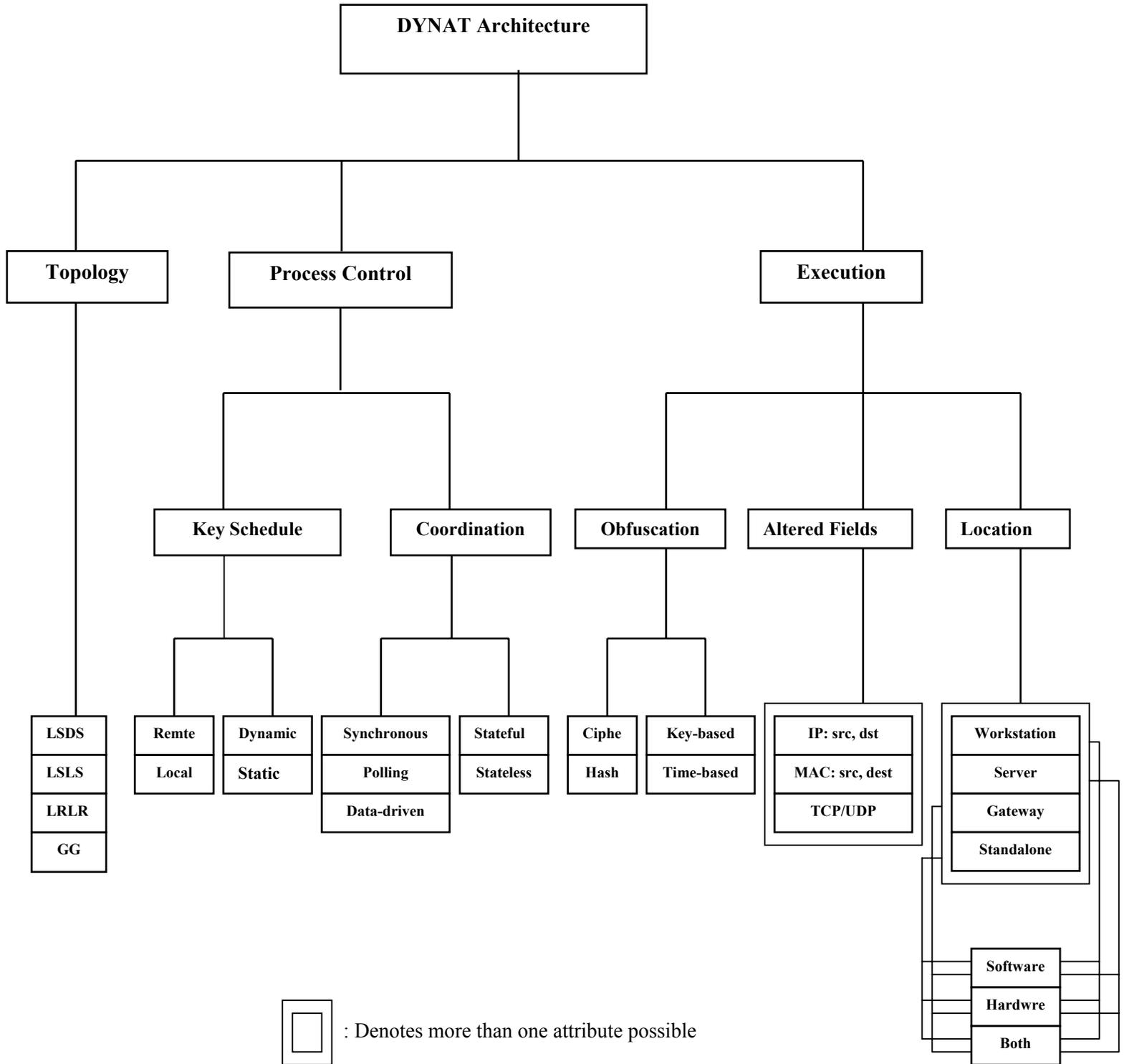
[KA98c] S.Kent, R.Atkinson, *IP Encapsulating Security Payload (ESP)*, IETF RFC 2406, Nov 1998.

Appendix A

Decision Tree

A decision tree can be used to determine which type of DYNAT process is compatible with a particular system or network. After traversing each branch of the tree, one can compile a set of keywords that describes the DYNAT implementation in question. For example, there are four main variations of topologies for DYNAT: LAN Segment to Distant Server (LSDS), Distant Gateway to Distant Gateway (DSDS), LAN Segment to LAN Segment (LSLS), and LAN Gateway to LAN Gateway (LGLG). Each of these will also have an associated listing of characteristics that describe the DYNAT's "Process Control" and "Execution". Process control is made up of the scheduling of the key that encodes the packet header as well as the coordination of when and by what mechanism the header is changed. The "Execution" category is meant to convey the "how" of the DYNAT process. It includes the method by which the header fields are encoded, exactly which fields are changed, and where on the computer (application level or hardware level) and on the network (workstation, server, gateway, or other device) the DYNAT code resides. From this decision tree, a person will be able to categorize his or her DYNAT system. Also, the tree format can be beneficial when implementing a DYNAT protocol by gaining a better understanding of the different attributes that make up this network technology.

DYNAT Decision Tree



Appendix B

The overall guiding principle on how a particular Dynat implementation is selected will be based on an existing customer architecture. The assumption is that customer's will have an installed network base on a pre-defined architecture. As a consultant the advice to rebuild an existing network is unreasonable. The starting point for analyzing the installation of a Dynat protocol will be built around the customers network architecture. With this in mind the following is a description of some potential approaches for using a Dynat, based on the attribute of the Dynat protocol and its residing architecture. This appendix is not structured to be reviewed serially by the reader. Only the topology of interest needs to be consulted.

1. Topology A

Architecture type: Clint to Server
Topology Implementation: LAN segment Local Server (LSLS)

Attribute Selection **Process control type:**
Synchronous, Time based polling, or Packet/Frame
Field Obfuscation:
IP and MAC layer
Key type and Distribution:
RSPP, LSPS
Dynat Code Type:
Software, Hybrid or Hardware

Implementation Discussion

A. Topology

The LAN segment Local Server (LSLS) topology allows LAN segment isolation for end users and selected services. If a router is used to interconnect other local or distant segments, user input and output filters could be implemented to help characterize traffic entering or exiting the segment.

This implementation allows a means of isolating the Dynat protocol to help heighten the security present on the operation LAN. It is important to note that all traffic exiting or entering this LAN segment may not be afforded all the standard protection approaches used to filter I/O traffic. This occurs because of the incompatibility that may exist between the selected Dynat process and common Firewall or proxy services. *See Performance Degradation section* for possible implementation problems.

B. Process Control Type

A distributed synchronous approach to process control allows the local workstations and servers to work independently of any external control function. This prevents any one node from becoming a central point of failure. The liability for this implementation is its reliance on a central clock. The synchronization of each individual host becomes critical because the change

(Dynat) algorithm is associated with a central clock. This process can be relatively efficient as long as its implementation limits the number of participating nodes.

The time based polling technique is dependent on the proper configuration and operation of a Dynat controller. This controller is normally responsible for initializing all participation nodes by transferring or referencing the keying material. Because the workstations and servers are no longer required to synchronize their operating clocks this implementation is more robust than its synchronous counterpart. The primary liability lies in its reliance on the controller node for Dynat control. If this node becomes incapacitated for any reason e.g. node failure, configuration management, or adversarial attack, all participation nodes will be unable to communicate with each other using the Dyant protocol.

C. Field Obfuscation

The fields selected from the communication protocol stack may vary (see section: *Field Obfuscation* in the body of this report for a list of selectable fields.) In general, the more fields that participate in the obfuscation the more difficult it is for an adversary to make any sense of the technique. It is important to note that to prevent an adversary from identifying a server on this segment, the MAC and IP layer must participate in the obfuscation. This is important because if the IP address is solely used to provide obfuscation, the mapping of the MAC address to its destination node can be accomplished. Servers, and even gateway nodes can be identified solely by the traffic originating and leaving their interfaces. If the MAC address is not Dynated, it could easily be correlated with its associated service. Once this is known, a denial of service could be launched using the destination address of the identified node. Also the general statement that the more fields that are obfuscated the more secure the protocol is not intended to be a blanket statement. If the obfuscation process is implemented solely in software then operating system performance becomes a more critical issue. *See Dynat Code Type.*

D. Key type and Distribution

As previously discussed in section *Key types and Distribution*, there are many different approaches to Dynat key construction and distribution. When using the synchronous type of process control, the type of key used will probably be any of the local variants. This is because adding a remote variable to an obfuscation engine that is being synchronized by a common clock across many nodes becomes an enormous task. Using the LSPS key allows all nodes to become independent of a clock source. This implementation provides a very robust approach to the obfuscation mechanism. There is no clock to reference, and thus, there are no synchronization problems. It becomes critical that the available key on each participating node is properly protected. Once the key is loaded into the obfuscation engine, it should be provided with proper user protections including login passwords and protected screen savers. And if possible, the key should be encrypted while being stored locally.

The LSPD key approach has the advantage of constant change. If a single key is compromised, its value is limited to its expiration, which in most cases, is short lived. In this case the protection of a single key is not the issue, but rather the process of creating the key becomes important. If an adversary were able to analyze the process then all future key generation would be jeopardized.

The LMPS&D key creates a key from two distinct parts: a static part, shared by all nodes, and a dynamic part that is referenced by all nodes. The obfuscation engine needs both parts to generate a properly formatted Dynat packet. If an adversary is able to obtain only one of these parts, he would be unable to decode any Dynat sessions, although if one of these parts were compromised it reduces the overall work that is needed to overcome the rest of the key. Although this combination key seems more difficult to break it does introduce the difficult task of synchronizing the dynamic process among many nodes. An example implementation of the LMPS&D scheme uses the University of California Berkeley (UCB) crypt program to obfuscate the destination IP address and destination port. This program is an 8-bit symmetric block encryption algorithm. For a class A address the 8 bit field from the host part of the IP address and the 16 bit field from the destination port field are copied out of the data gram and used as the “data” input to this block encryptor. The key is comprised from a secret seed (that is shared by the Dynat client and the Dynat server) and the time, based in seconds. This creates the new fields that are inserted back into the packet with a new calculated CRC. The Dynat Client and the Dynat server must be synchronized to create a properly shared key.

The only question that arises with this encryption scheme involves subnetting across 8 bit boundaries. For example, if a host resides on the following network 4.22.160.64 and is subnetted with 26 bits, the host portion of the address that is manipulated by Dynat contains only 6 bits. What happens if only 6 bits of host ID is added to 16 bits of port ID to create a total of bits that are not a multiple of 8? Padding could be added to resolve this problem but this becomes another complexity in the overall scheme.

The RSPP key implementation is more unique than other approaches. The key is not a key at all but is comprised of a control signal that is sent to all participating nodes. The control pulse could then increment a counter or pointer that would be used as an input value to the obfuscation engine. Of course if the control pulse were not delivered at the proper time to the participating end host that host would be unable to communicate with any other hosts.

The key can also be a table of network protocol fields that are used in the obfuscation. The network protocol fields are obtained by subsequent iterative cycles of a obfuscation algorithm. The new values can be created locally on each individual host or can be done locally at a controller node and distributed to each participating end host as the control signal. If this process is done locally on each host, then the new values are not presented across the network. It is therefore important to protect the obfuscation algorithm from being compromised on each host. If the obfuscation algorithm it is only resident on the controller node then this node must be physically protected with the new obfuscation out put values need to be protected while “in-flight” to each participating end host

E. Dynat Code Type

The best two choices for Dynat code implementation in the LAN segment Local Server topology (LSLS) are the hardware and hybrid based approaches. The hardware approach allows all the obfuscation operation to be inserted into logic located on participating nodes. The advantages of this approach allow for very efficient use of Dynat manipulations. This can reduce the burden placed on the operating systems when hosting the Dynat process, or depending on construction, can eliminate the need for operating system intervention. The liabilities encountered with this approach include the inability to change a particular process or filter without a redesign of the on-board logic, although depending on technology chosen, it can be possible to change a process or filter using flash programmable logic. But since all the information about the Dynat process is

stored in hardware, allows for an adversary to concentrate resources on the card and analyze its construction in detail.

A better implementation is found in the hybrid approach. The key can be software based and follow any of the pre-described choices found in the *Key type and Distribution section*. Since this approach allows the Dynat process to be changed by incorporating different filters that are downloaded into firmware, it allows individual hosts to be tailored based on a users profile. It also can be constructed to prevent review and tampering with the obfuscation engine and associated filters by allowing the logic to be nulled when power is no longer applied to the device.

A strictly software solution, although easier to distribute and use, causes a heavy tax to be levied on the operating system. Software solutions normally require the pre and post manipulations of packets that flow up and down the operating stack. This packet processing occurs independent of the “rate of change” of the keying process in the Dynat protocol. This means that as packet throughput increases, so does the processing power needed to manipulate each packet.

2. Topology B

Architecture type: Clint to Server
Topology Implementation: LAN segment Distant Server (LSDS)

Attribute Selection **Process control type:**
 Synchronous, Time based polling, Packet/Frame
 Field Obfuscation:
 IP and MAC layer
 Key type and Distribution:
 LSRS, RSPS or RSPP
 Dynat Code Type:
 Software, Hybrid or Hardware

Implementation Discussion

A. Topology

The topology used for the deployment of this Dynat approach includes two Ethernet segments segregated by a Router. The first Ethernet LAN segment contains clients that participate in the Dynat process. The clients must traverse a router to gain access to the second LAN segment, which contains the appropriate servers. The router participates in the Dynat process or can be front-ended with a stand-alone device that performs the necessary translations. The “distant server” in this example can also be connected across the Internet. This demonstrates that the Dynat process doesn’t inhibit routed transport across the Internet. See the Architecture Support section figures 3 and 3.A for a view of these configurations.

Advantages of this architecture:

- Routers allow additional filtering capability for all incoming packets. For example all source addresses can be checked for proper subnet origination.
- Dynat located on router allows a “concentration effect” for analysis of valid destination addresses to services located on adjacent LAN. Effective for verification and reporting.

- Router can be located in a separate physical location for harden physical layer protection.

Disadvantages:

- Routing packets between subnets induces additional processing delays and single point of failure vulnerability.
- Adds hardware to system, which incurs additional maintenance, configuration, and costs.
- Server LAN must be physically isolated from Client LAN (Server LAN is operated in the clear)

B. Process Control Type

A distributed synchronous approach to the process control allows the local workstations and servers to work independently of any external control function. This prevents any one node from becoming a central point of failure. The liability for this implementation is the reliance on a central clock. The synchronization of each individual host becomes critical because the change (Dynat) algorithm is associated with the central clock. This process can work relatively efficiently as long as its implementation limits the number of participating nodes.

The time based polling technique is dependent on the proper configuration and operation of a Dynat controller. This controller is normally responsible for initializing all participation nodes by transferring or referencing the keying material. Because the workstations and servers are no longer required to synchronize their operating clocks, this implementation is much more robust than its synchronous counterpart. The primary liability occurs in the reliance on the controller node for Dynat control. If this node becomes incapacitated for any reason e.g. node failure, configuration management, or adversarial attack, all participation nodes will be unable to communicate with each other using the dyant protocol.

The packet/frame based process control technique for hosts on a LAN segment must include a mechanism to keep track of the number of packets or frames that have been transmitted on any given pair of end node to end node connections. Since the servers do not participate in the Dynat process, the interconnecting Router or Interface device to the Router would have to keep a record of all packets or frames on each individual client/server connection to be able to implement this approach. If the Router or Interface device implementing this approach were temporarily impaired all connections for each client would lose synchronization and be lost. This technique becomes unwieldy to implement. See section Dynat Process Control for examples of this technique.

C. Field Obfuscation

The fields selected from the communication protocol stack may vary (see section: *Field Obfuscation* in the body of this report for a list of selectable fields.) In general, the more fields that participate in the obfuscation the more difficult it is for an adversary to make any sense of the technique. It is important to note that to prevent an adversary from identifying a server on this segment, the MAC and IP layer must participate in the obfuscation. This is important because if the IP address is solely used to provide obfuscation, the mapping of the MAC address to its destination node can be accomplished. Servers, and even gateway nodes can be identified solely by the traffic originating and leaving their interfaces. If the MAC address is not Dynated, it could

easily be correlated with its associated service. Once this is known a denial of service could be launched using the destination address of the identified node. Also the general statement that the more fields that are obfuscated the more secure the protocol is not intended to be a blanket statement. If the obfuscation process is implemented solely in software then operating system performance becomes a more critical issue. *See Dynat Code Type.*

One approach to obfuscate the MAC address is to implement this obfuscation process in a stand-alone device. This device could be implemented to capture ARP requests by clients looking for the Gateways MAC addresses and respond with an ever-changing ARP response. All the clients could run in promiscuous mode with their ARP response mechanisms disabled. This dummy ARP responder would generate random ARP addresses for all request thus satisfying the clients need to have a MAC destination address. Since all clients and servers have their NIC cards in promiscuous mode they would all take in the subsequent packets, only packets addressed to the appropriate end node would be processed. Of course the major draw back to this implementation is the heavy load of traffic that would be present at “every” NIC interface of “every” node, and the dummy ARP responder must then provide the appropriate translation to the Router before forwarding the Frame. ARP cache mechanisms in each host must also be suspended to prevent Frame deliveries to non-valid addresses.

D. Key Type and Distribution

As previously discussed in section *Key types and Distribution*, there are many different approaches to Dynat key construction and distribution. When using the synchronous type of process control, the type of key used will probably be any of the Local variants. This is because adding a remote variable to an obfuscation engine that is being synchronized by a common clock across many nodes becomes an enormous task.

Using the LSPS key allows all nodes to become independent of a clock source. This implementation provides a very robust approach to the obfuscation mechanism. There is no clock to reference, and thus, there are no synchronization problems. It become critical that the key that is available on each participating node is properly protected. Once the key is loaded into the obfuscation engine, it should be provided with proper user protections including login passwords and protected screen savers. And if possible the key should be encrypted while being stored in any file. Another key that could be implemented with the synchronous type of process control is the Local Single Part Dynamic (LSPD).

The LSPD key approach has the advantage of constant change. If a single key is compromised its value is limited to its expiration, which in most cases, is short lived. In this case the protection of a single key is not the issue, but rather the process of creating the key becomes important. If an adversary were able to analyze the process, then all future key generation would be jeopardized.

The LMPS&D key creates a key from two distinct parts: a static part, shared by all nodes, and a dynamic part that is referenced by all nodes. The obfuscation engine needs both parts to generate a properly formatted Dynat packet. If an adversary is able to obtain only one of these parts, he would be unable to decode any Dynat sessions, although if one of these parts were compromised it reduces the overall work that is needed to overcome the rest of the key. Although this combination key seems more difficult to break, it does introduce the difficult task of synchronizing the dynamic process among many nodes.

When using the Polling approach to process control the Remote Single Part Polling (RSPP) key is the most easiest to implement. This is because the key does not necessarily need to be comprised of a value. It could simple be some sort of control pulse that is delivered from the controller to each participating node. This control pulse could then increment a counter or pointer that would be used as an input value to the obfuscation engine. Of course if the control pulse were not delivered at the proper time to the participating end host that host would be unable to communicate with any other hosts.

The Local Single Part Static (LSPS) key could also be used to enable the obfuscation engine. This key would be locally administrated for each participating node and would be used in conjunction with control information form the controller. This allows for two different pieces of data needed to drive the obfuscation process.

The Remote Single Part Static (RSPS) key would be distributed from a central location making it easier to administer to each participating node. It would be downloaded prior to the initiation of the polling sequence. As with any over-the-network key delivery mechanism it should be protected from snooping.

E. Dynat Code type

The best two choices for Dynat code implementation in the LAN Segment Distant Server (LSDS) topology is the hardware or hybrid based approach. The hardware approach allows all the obfuscation operation to be inserted into logic located on participating nodes. The advantages of this approach allows for very efficient use of Dynat manipulations. This can reduce the burden placed upon the operating systems when hosting the Dynat process, or depending on construction, can eliminate the need for Operating system intervention. The liabilities encountered with this approach have to do with the inability too change a particular process or filter without a redesign of the on-board logic, and since all the information about the Dynat process is stored in hardware allows for an adversary to concentrate resources on the card and analyze its construction in detail.

A better implementation is found in the hybrid approach. The key can be software based and follow any of the pre-described choices found in the *Key type and Distribution section*. Since this approach allows the Dynat process to be changed by incorporating different filters that are downloaded into firmware, it allows individual hosts to be tailored based on a users profile. It also can be constructed to prevent the review and tampering with the obfuscation engine and associated filters by allowing the logic to be dispersed when power is no longer applied to the device.

A strictly software solution, although easier to distribute and use, causes a heavy tax to be leveled on the operating system. Software solutions normally require the pre and post manipulations of packets that flow up and down the operating stack. This packet processing occurs independent on the “rate of change” of the keying process of the Dynat protocol. These means as packet throughput increases so does the processing power needed to manipulate each packet.

3. Topology C

Architecture type: Distribution
Topology Implementation: Local Router to Local Router

Attribute Selection	<p>Process control type: Synchronous or Packet/frame</p> <p>Field Obfuscation: TCP, UDP, IP, and MAC layer</p> <p>Key type and Distribution: Combination, LSPS or LSPD</p> <p>Dynat Code Type: Software, Hybrid or Hardware</p>
---------------------	---

Implementation Discussion

A. Topology

In the Router-to-Router implementation, the link between the routers carries the protocol obfuscation. Unlike previous topologies, the Dynat process is not implemented on any client or server hosts. This approach attempts to protect information about the routed traffic between the two local segments, and removes any processing burden from the clients and transfers it to the Routers.

Concentrating resources to tightly dedicate these two routers for Dynat operations has some merit. For example, if a synchronous process control approach is used to control the Dynat process, it becomes easier to maintain this synchronous time frame on two machines instead of many. Also, implementing an Intrusion Detection mechanism on the Routers becomes more practical (See the *IDS Integration* section for more detail). Also, additional packet filtering can be enabled in the form of Access Control Lists (ACL's) that allow tighter control of data throughput.

A limitation to this architectural approach is that it does not address the insider threat on the local LAN segments. It assumes an adversary with access to the link between the routers cannot do sniffing on the client or sever segments. This approach also aggregates many traffic flows introducing a single point of failure.

Note it is not necessary to incorporate the Dynat process code into the routers themselves. The Dynat process can be inserted in standalone devices that sit between the routed links. This location is important because if the Dynat devices where in front of each router, the router would be unable to perform host level filtering. If TCP and UDP port obfuscation were deployed, the router would be unable to filter on applications.

B. Process Control Type

A distributed synchronous approach to the Dynat process control allows the routers or standalone Dynat devices to work independently of any external control function. In the LAN views, this would prevent any one node from becoming a central point of failure. Because there are only two nodes in this topology participating in the Dynat process, the use of this process control type to prevent a central point of failure, has no merit.

The liability for this implementation is the reliance on a central clock. The synchronization of each individual router or standalone device becomes critical because the change (Dynat) algorithm is associated with the central clock. Although this synchronization process becomes difficult when there are many nodes participating, it is a lot more effective when only two nodes are involved.

The time based polling technique is dependent on the proper configuration and operation of a Dynat controller. This controller is normally responsible for initializing all participation nodes by transferring or referencing the keying material. Because the Routers are standalone Dynat devices are no longer required to synchronize their operating clocks this implementation is a lot more robust than its synchronous counter part. Although this approach is more robust than the distributive synchronous approach it introduces the need for an additional control node. The primary liability occurs in the reliance on the controller node for process control. If the controller node becomes incapacitated for any reason e.g. node failure, configuration management, or adversarial attack, the participating router nodes will be unable to communicate with each other using the Dyant protocol.

It is also not clear on where in this topology the Controller node would be located. If it is located on the link between the routers then it is open to adversarial attack. See the *Architecture Support* section for a view of this approach

The packet/frame based process control technique fits more readily with the router-to router approach to the Dynat implementation. The reason is the router only needs to process one data stream per link. This isolated link-by-link processing allows for a concentrated approach to the obfuscation process. Keeping synchronization between the end devices becomes easier because there are only two participating devices. Because this implementation is concentrated between a few LAN segments the process becomes more deterministic. This is because all destination network addresses that will flow across the network link are known beforehand, and their numbers are small. This allows for ease of design when using a table, index, or register in the obfuscation process.

If this process is co-located on the routers, it will introduce additional processing and memory burdens for the router. It is also not clear when using the packet or frame synchronization approach, how the recovery mechanism would work when the nodes fall out of synchronization. Or even more importantly, how the end nodes would detect a synchronization failure.

C. Field Obfuscation

The fields selected from the communication protocol stack may vary (see section: *Field Obfuscation* in the body of this report for a list of selectable fields.) In general the more fields that participate in the obfuscation the more difficult it is for an adversary to make any sense of the technique. It is important to note that to prevent an adversary from identifying the router ports on this inter-connected link the MAC field must also be included in the obfuscation process. This identification occurs because packets are delivered hop by hop with each hop being addressed by its MAC address. MAC obfuscation on this configuration prevents an adversary from addressing packets to the router ports. Also to increase the effort of an adversary that maybe interested in performing traffic analysis of the types of data that may traverse this link it is important to add TCP or UDP port obfuscation to the Dynat process.

The general statement that the more fields that are obfuscated the more secure the protocol is not intended to be a blanked statement. If the obfuscation process is implemented solely in software then operating system performance becomes a more critical issue. See the *Dynat Code Type section*.

If the router to router obfuscation method is implemented in hardware or even in a hybrid approach the concern for processing delay is reduce allowing the obfuscation to take full advantage of all needed and available fields.

D. Key Type and Distribution

A combination key (CMPS&D) uses a local component and a remote component that are “combined” to form the overall key. The local component of this key is normally static but can be implemented in a dynamic fashion. The remote component of the key is sent by a control function that distributes this part of the key to all participating nodes. Both parts are combined on the participating nodes to drive the obfuscation engine. Combining these two parts can take multiple forms. A static key can be installed locally on each router node and have the dynamic part of the key downloaded from a network controller. The frequency of downloading this portion of the key can be set by the administrator, and can vary from seconds to days. This implementation prevents an adversary from gaining permanent information about both co-located keys.

A local implementation of the combination key allows for the static part of the key and the dynamic portion of the key to be always resident on the routers. The dynamic portion of the key can be locally changed based on a clock or incrementing index. This prevents the dynamic portion of the key from being seen or stolen when being sent to the router.

Any of the Local Single part Dynamic or Static choices of keys can be easily implemented using this topology. Because there are only a few nodes participating in this process, key management is easier, especially if the key type is dynamic. The synchronization of the keys to a single clock source is easier to maintain when there are only a few nodes participating in the synchronization process.

If a packet-by-packet change rate is applied then the key maybe implemented as an index that is incremented based on a rate that is related to the packet rate. This type of implementation can be vulnerable to compromise if the elements that comprise the incremental array are discovered. Also if the index offsets of the participating nodes fall out of sync, detection of this state becomes crucial because many packets will not be addressed properly and would not be processed.

E. Dynat Code type

The Dynat code type for this implementation will probably be of the hybrid or hardware type. Although the software approach is feasible, it is not very practical considering the link speeds available in modern networking. Packet field manipulation that is not built into hardware creates bottlenecks when attempting to process these fields at line speeds.

A truly hardware approach will have the entire obfuscation process contained in hardware. These means that the key distribution process must take place during an initialization process of the hardware.

4. Topology D

Architecture type: Distribution
Topology Implementation: Gateway-to-Gateway

Attribute Selection **Process control type:**
 Packet/frame or Time based polling
 Field Obfuscation:
 Must include MAC layer
 Key type and Distribution:

LSPS, LSPD, LMPS&D
Dynat Code Type:
Hybrid or Hardware

Implementation Discussion

A. Topology

The Gateway-to-Gateway implementation is similar to the Router-to-Router approach except the interconnecting Gateway link becomes the public medium (i.e. Internet.) This connection implies there are many intermediate nodes that exist between the Gateways that are not participating in the Dynat protocol.

This obfuscation does not prevent packets from being routed across the Internet because only the host portion of the IP address can be manipulated. It is important to note when choosing an address mask to determine network to host delineation, the participating routing protocols are able to distinguish address assignments outside of standard byte boundaries. Gateways that are able to participate in variable sub length masking (VSLM) can provide an aggregation to the network routes advertised, and can reduce the size of the routing tables. In some cases relating to the Dynat process this can reduce the host ID portion of the address, because of the subnet advertisement capability, thus decreasing the obfuscation space of the Dynat protocol. This can make an adversary's work at predicting the obfuscation address sequence less difficult.

Unlike the previous topologies the Dynat process is not implemented on any client or server hosts. This approach attempts to protect information about the routed traffic between the two distant Gateways.

This removes any processing burden from the clients and transfers it to the Gateways.

Concentrating resources to tightly dedicate these two gateways for Dynat operations has some merit. For example, if a synchronous process control approach is used to control the Dynat process it becomes easier to maintain this synchronous time frame on two machines instead of many. Also implementing an Intrusion Detection mechanism on the Gateways becomes more practical (See the *IDS Integration* section for more detail). Also, additional packet filtering can be enabled in the form of Access Control Lists (ACL's) that allow for tighter control of data throughput.

A limitation to this architectural approach is that it does not address the insider threat on the local networks attached to the Gateways. It assumes an adversary has access to the public link between the Gateways but cannot do sniffing on the local networks. This approach also aggregates many traffic flows introducing a single point of failure.

Note that it is not necessary to incorporate the Dynat process code into the Gateways themselves. The Dynat process can be inserted in standalone devices that sit between the routed links. This location is important because if the Dynat devices were in front of each Gateway, the Gateway would be unable to perform host level filtering. If TCP and UDP port obfuscation were deployed the Gateway would be unable to filter on applications.

B. Process Control Type

A distributed synchronous approach to Dynat process control allows the Gateways or standalone Dynat devices to work independently of any external control function.

The liability for this implementation is on the reliance on a central clock. The synchronization of each individual Gateway or standalone device becomes critical because the change (Dynat) algorithm is associated with the central clock. Although this synchronization process becomes difficult when there are many nodes participating, it is a lot more effective when only two nodes are involved.

The packet/frame based process control technique can only be implemented if a VPN is constructed between the Gateways. The need for the VPN's arises because the number of packets/frames sent between the participating nodes must be known. Packets originating from some other source but terminating at the Gateways would not be accountable and thus undermine the synchronization process that must exist between the Gateways

A liability to this approach occurs if the obfuscation process is co-located on the Gateways. This introduces additional processing and memory burdens for the Gateway.

It is also not clear when using the packet or frame synchronization approach how the recovery mechanism would work when the nodes fall out of synchronization. Or more importantly how the end nodes would detect a synchronization failure.

It seems quite possible that if an adversary would destroy a packet en route between the gateways that the packets "non-arrival" would cause a synchronization problem. Even if the process between the gateways allowed for a connection-orientated signaling approach to managing interactions, keeping track of individual packets could become process intensive.

C. Field Obfuscation

The fields selected from the communication protocol stack may vary (see section: *Field Obfuscation* in the body of this report for a list of selectable fields.) In general the more fields that participate in the obfuscation the more difficult it is for an adversary to make any sense of the technique. Unlike the similar Router-to-Router implementation the Gateway-to-Gateway approach cannot take advantage of using the MAC address space for obfuscation. This is because there are many intermediate nodes that exist between the Gateways that are not participating in the Dynat process. And since packets are delivered hop by hop using the MAC destination address of the "next hop" Router, this field cannot be obfuscated when intermediate nodes are not participating in the Dynat process.

This limitation does not prevent higher layers in the protocol stack from being obfuscated. If, for example, there is a need to prevent an adversary snooping the inter-connected link between the Gateways from determining the services that are being supported, then obfuscating the TCP or UDP port numbers on the packet streams between the Gateways can prevent this information from being discovered.

The general statement that the more fields that are obfuscated the more secure the protocol is not intended to be a blanket statement. If the obfuscation process is implemented strictly in software then operating system performance becomes a more critical issue. See *Dynat Code Type*.

D. Key Type and Distribution

A combination key (CMPS&D) uses a local component and a remote component that are "combined" to form the overall key. The local component of this key is normally static but can be implemented in a dynamic fashion. The remote component of the key is sent by a control function that distributes this part of the key to all participating nodes. Both of these parts are

combined on the participating nodes to drive the obfuscation engine. Combining these two parts can take multiple forms. A static key can be installed locally on each Gateway node and have the dynamic part of the key downloaded from a network controller. The network controller node can even take the form of a Certificate Authority (CA). The frequency of downloading this portion of the key can be set by the administrator and can vary from seconds to days. This implementation prevents an adversary from gaining permanent information about both co-located keys.

A local implementation of the combination key allows the static part of the key and the dynamic portion of the key to be always resident on the routers. The dynamic portion of the key can be locally changed based on a clock or incrementing index. This prevents the dynamic portion of the key from being seen or stolen when being sent to the router.

Any of the Local Single part Dynamic or Static choices of keys can be easily implemented using this topology. Because there are only a few nodes participating in this process key management is easier, especially if the key type is dynamic. The synchronization of the keys to a single clock source is easier to maintain when there are only a few nodes participating in the synchronization process.

If a packet-by-packet change rate is applied then the key maybe implemented as an index that is incremented based on some rate that is relational to the packet rate. This type of implementation can be vulnerable to compromise if the elements that comprise the incremental array are discovered. Also if the index offsets of the participating nodes fall out of synch, detection of this state becomes crucial because many packets will not be addressed properly and would not be processed.

E. Dynat Code type

The Dynat code type for this implementation will probably be of the hybrid or hardware type. Although the software approach can be used if the VPN that is created between the Gateways limits the amount of traffic transmitted through it. Otherwise, it is not very practical implementing the Dynat code in software considering the link speeds available in modern networking. Packet field manipulation that is not built into hardware creates bottlenecks when attempting to process these fields at line speeds.

A truly hardware approach will have the entire obfuscation process contained in hardware. These means that the key distribution process must take place during an initialization process of the hardware.

5. Topology E

Architecture type: Clint to Server
Topology Implementation: LAN segment to LAN segment

Attribute Selection **Process control type:**
Packet/frame or Time based polling
Field Obfuscation:
Must include MAC layer
Key type and Distribution:
LSPS, LSPD, RSPP, Combination
Dynat Code Type:

Implementation Discussion

A. Topology

One slight variation to all previously described Dynat topologies involves two or more distinct segments implementing the Dynat protocol on all hosts and maintaining this obfuscation across interconnected Routers or Gateways. This implies that the distinct Dynat processes must provide coordination between their interconnected segments. Routers and gateways must also participate in the Dynat protocol. This participation may be located on each Router or Gateway or in the form of an external “stand alone” interface. This implementation is the most comprehensive approach providing obfuscation for end nodes on each LAN segment, and also providing an obfuscated stream between segments.

Although this approach seems the most protective, there are some limitations that should be mentioned.

When obfuscating a packet stream through a Router or Gateway, it is important to realize that any type of Network Address Translation (NAT) or Port Address Translation (PAT) function available within the Router or Gateway must be disabled. The reason to disable NAT and PAT is that these functions require that state be maintained for individual connections. The state information contains IP address and Port numbers that are used to reference these connections. When implementing a Dynat protocol one or both of these fields will continually be changed making state information recording impossible.

This problem can even extend itself to application type proxies that are used to monitor content information from packet streams originating and terminating within a LAN segment.

B. Process Control Type

A distributed synchronous approach to the process control allows local workstations and servers to work independently of any external control function. This prevents any one node from becoming a central point of failure. The liability for this implementation is its reliance on a central clock. The synchronization of each individual host becomes critical because the change (Dynat) algorithm is associated with a central clock. This process can be relatively efficient as long as its implementation limits the number of participating nodes. This technique can be implemented in the LAN segment to LAN segment topology with each individual LAN maintaining its own synchronization. There is not necessarily a need to maintain synchronization across different LAN segments. But some other means of providing information about the separate Dynat realms must be provided.

The time based polling technique is dependent on the proper configuration and operation of a Dynat controller. This controller is normally responsible for initializing all participation nodes by transferring or referencing the keying material. Because the workstations and servers are no longer required to synchronize their operating clocks, this implementation is more robust than its synchronous counterpart. The primary liability lies in its reliance on the controller node for Dynat control. If this node becomes incapacitated for any reason e.g. node failure, configuration management, or adversarial attack, all participation nodes will be unable to communicate with each other using the Dynat protocol.

How the controller node is implemented can vary. For example, there can be one controller node that provides the obfuscation control across multiple LAN segments, or each LAN segment can

contains its own controller. Regardless of the controller implementation it will always be important to be able to coordinate the Dynat realms across each individual LAN segment.

The packet/frame-based approach is unruly to implement within this topology. It lends itself more to the Router-to-Router or Gateway-to-Gateway topologies.

See *Dynat Process Control* section in the main body of this report for a discussion of possible packer/frame based approaches

C. Field Obfuscation

The fields selected from the communication protocol stack may vary (see section: *Field Obfuscation* in the body of this report for a list of selectable fields.) In general, the more fields that participate in the obfuscation the more difficult it is for an adversary to make any sense of the technique. It is important to note that to prevent an adversary from identifying a server on this segment, the MAC and IP layer must participate in the obfuscation. This is important because if the IP address is solely used to provide obfuscation, the mapping of the MAC address to its destination node can be accomplished. Servers, and even gateway nodes can be identified solely by the traffic originating and leaving their interfaces. If the MAC address is not “Dynated”, it could easily be correlated with its associated service. Once this is known, a denial of service could be launched using the destination address of the identified node. Also the general statement that the more fields that are obfuscated the more secure the protocol is not intended to be a blanket statement. If the obfuscation process is implemented solely in software then operating system performance becomes a more critical issue. See *Dynat Code Type*.

One approach to obfuscate the MAC address is to implement this obfuscation process in a stand-alone device. This device could be implemented to capture ARP requests by clients looking for the Gateways MAC addresses and respond with an ever-changing ARP response. All the clients could run in promiscuous mode with their ARP response mechanisms disabled. This dummy ARP responder would generate random ARP addresses for all requests, thus satisfying the client’s need to have a MAC destination address. Since all clients and servers have their NIC cards in promiscuous mode they would all take in the subsequent packets. Only packets addressed to the appropriate end node would be processed. Of course the major draw back to this implementation is the heavy traffic load that would be present at “every” NIC interface of “every” node, and the dummy ARP responder must then provide the appropriate translation to the Router before forwarding the Frame. ARP cache mechanisms in each host must also be suspended to prevent Frame deliveries to non-valid addresses.

Another note about MAC address obfuscation; If the obfuscation function is extended across distant LAN segments, the MAC obfuscation process will be suspended while traversing Router links. This must occur because the “routing process” is implemented in a hop-by-hop fashion with the MAC destination addresses being changed to the valid MAC address of the next hop router.

D. Key Type and Distribution

As previously discussed in section *Key types and Distribution*, there are many different approaches on how the Dynat key is constructed and distributed. When using the synchronous type of process control, the type of key used will probably be any of the Local variants. This is because adding a remote variable to an obfuscation engine that is being synchronized by a common clock across many nodes becomes an enormous task. Using the LSPS key allows all nodes to become independent of a clock source. This implementation provides a very robust

approach to the obfuscation mechanism. There is no clock to reference, and thus, there are no synchronization problems. It becomes critical that the available key on each participating node is properly protected. Once the key is loaded into the obfuscation engine, it should be provided with proper user protections including login passwords and protected screen savers. And if possible the key should be encrypted while being stored in any file.

The LSPD key approach has the advantage of constant change. If a single key is compromised, its value is limited to the time available before its expiration, which in most cases is short lived. In this case the protection of a single key is not the issue, but rather the process of creating the key becomes important. If an adversary were able to analyze the process then all future key generation would be jeopardized.

The LMPS&D key creates a key from two distinct parts: a static part, shared by all nodes, and a dynamic part that is referenced by all nodes. The obfuscation engine needs both parts to generate a properly formatted Dynat packet. If an adversary is able to obtain only one of these parts, he would be unable to decode any Dynat sessions, although if one of these parts were compromised it reduces the overall work that is needed to overcome the rest of the key. Although this combination key seems more difficult to break, it does introduce the difficult task of synchronizing the dynamic process among many nodes.

The RSPP key implementation is more unique than other approaches. The key is not a key at all but is comprised of a control signal that is sent to all participating nodes. This control pulse could then increment a counter or pointer that would be used as an input value to the obfuscation engine. Of course if the control pulse were not delivered at the proper time to the participating end host, that host would be unable to communicate with any other hosts.

The key could also be a table of network protocol fields used in the obfuscation. The network protocol fields are obtained by subsequent iterative cycles of an obfuscation algorithm. The new values can be created locally on each individual host or can be done locally at a controller node and distributed to each participating end host as the control signal.

If this process is done locally on each host, then the new values are not presented across the network. It is therefore important to protect the obfuscation algorithm from being compromised on each host. If it is only resident on the controller node then this node must be physically protected while the new obfuscation values must be protected while “in-flight” to each participating end host.

E. Dynat Code type

A hardware approach allows all the obfuscation operation to be inserted into logic located on participating nodes. The advantages of this approach allows for very efficient use of Dynat manipulations. This can reduce the burden placed upon the operating systems when hosting the Dynat process, or depending on construction, can eliminate the need for operating system intervention. The liabilities encountered with this approach have to do with the inability to change a particular process or filter without a redesign of the on-board logic, and allows an adversary, who is able to obtain the hardware, to concentrate resources on the hardware and analyze its construction in detail. It is important to provide good physical protection to each participating node when the hardware approach is used.

A better implementation is found in the hybrid approach. The key can be software based and follow any of the pre-described choices found in the *Key type and Distribution section*. Since this approach allows the Dynat process to be changed by incorporating different filters that are

downloaded into firmware, it allows individual hosts to be tailored based on a users profile. It also can be constructed to prevent review and tampering with the obfuscation engine and associated filters by allowing the logic to be dispersed when power is no longer applied to the device.

A strictly software solution, although easier to distribute and use, causes a heavy tax to be levied on the operating system. Software solutions normally require the pre and post manipulations of packets that flow up and down the operating stack. This packet processing occurs independent of the “rate of change” of the keying process in the Dynat protocol. This means that as packet throughput increases, so does the processing power needed to manipulate each packet.

Appendix C

1. Information Assurance

The term “Information Assurance” refers to the ability of a system to protect the availability, confidentiality, integrity, reliability, and authenticity of the data. In this context, the following is a discussion of what information assurance attributes may be applied to the Dynat technology.

Availability of the data refers to the idea that the data is accessible to all authorized users at all times. Its unavailability maybe induced in either a physical or logical way.

The physical means that can be used to prevent timely delivery of data, such as the failure of critical network components, power disruptions, physical plant disruptions, either malicious or natural, is not associated with the discussion of the Dynat technology, and thus is beyond the scope of this discussion. The focus here is on logical obstructions that prevent the flow of data, and how a Dynat protocol can prevent or lessen the impact of these obstructions (more specifically, data denial through the use of denial-of-service attacks and address spoofing.) See the following *Adversarial Impact “Denial-of-Service”* section for a detail description of these types of attacks and the Dynat protocol response.

Confidentiality of information refers to the protection of data that allows only the intended recipient to be able to read the information. Most implementations of confidentiality rely on some form of encryption to prevent the disclosure of the information while the data is “in-flight” to its destination. Although standard forms of encryption can be inserted into the Dynat protocols to provide a means to add confidentiality to the data stream, the discussion here will concentrate on inherent forms of data confidentiality that exist in the Dynat protocols.

An adversary snooping the network and capturing packets for review, relies on a tool called a network analyzer to decode and succinctly catalog the captured packets. A network analyzer uses the standard protocol fields available in the different layers of the communication protocol stack to help in the arrangement and decoding of the incoming information. When these fields are manipulated by a Dyant protocol, the task of succinctly decoding each subsequent data packet becomes more difficult to automate without prior knowledge of the obfuscation technique. For example, In an IP session when the IP address and the identification field of the IP packet header between two communicating hosts have been obfuscated, it becomes very difficult to isolate the data flow with the use of the Network analyzer. Although the packets are sent in the clear, it becomes a laborious process to re-create the data sequence. This, in essence, provides a limited type of confidentiality to the data flow between the individual hosts by increasing the level of effort required to reassemble a packet stream.

Integrity of information refers to the ability of a system or mechanism to detect changes or modifications to an original message. Modern techniques implement integrity across a packet header and/or data field by creating a hash across the contents of the packet. This hash is based on a one-way function, and can detect any modifications to the original contents of the packet. The Dynat protocol itself provides no inherent form of integrity within the contents of a packet. If an Obfuscation technique is compromised, it is quite possible for a man-in-the-middle attack to be implemented within the confines of a Dynat protocol. For a Dynat protocol to be immune to packet manipulations, it must add additional protection, possibly in the form of a hash or digital signature, to the process. See the following *Adversarial Impact “Man-In-The-Middle”* section for a more detailed description of this form of attack and the Dynat response.

Reliability within the context of data communications refers to the ability of a communication system to provide consistent intended service over a large percentage of the time. The reliability of the data transmitted over this network is subject to the inter-connected network components of the system and the protocols that are used to provide the end host to end host communications. Communication protocols can provide a “reliable” facet to the data communications process. For example a somewhat noisy network link creating bit errors within a packet does not by itself prevent the communications between two communicating end nodes if the communications protocol is able to detect and retransmit the offended packets. The reliability of the packet communication process can still remain high in spite of occasional bit errors injected by the network link. The Dynat protocols and implemented techniques do not provide any protection that doesn’t already exist within the normal communications process.

Authenticity of data refers to its original conception and the binding of its author. Maintaining this relationship of data and associated author in modern network communications today is done with the use of public key encryption and a process called a digital signature. To create a digital signature a hash is created across the data. This hash is sometimes referred to as a message digest. This hash creates a one-way cryptographically strong series of bits that represent the original contents of the message. These bits are then encrypted with the private key of the author and sent along with the original message. The recipient is then able to verify the message content by decrypting the message digest with the author’s public key and comparing this output with the output of the received message’s hash. The Dynat protocols do not provide any additional means of authenticating data.

2. Adversarial Impact

This section addresses the ability of the Dynat protocols to prevent or reduce the adversarial impact to networks. The term “adversary” in the context of the following discussion refers to a hacker that is motivated by the extraction of unauthorized data, or disruption of network services.

The adversary has collaboration capability with other hackers, but limited financial resources. The Adversarial Impact refers to adversarial processes and techniques that are used to gather information about networks and services residing on them, and/or, to disrupt these services. Although there are many different techniques to disrupt services and/or extract unauthorized information from networks and the hosts residing on them, the majority of these techniques are contained in the following:

- Network Reconnaissance
- Man-In-The-Middle
- Denial-of-Service,

Traffic Analysis.

2.1 Network Reconnaissance

The term network reconnaissance refers to all techniques that are used to gather information about the following:

- Network segment IP range
- Active Hosts IP address on segment
- Files and Services Located on each Host
- Hosts Operating Systems

The above listed information is needed to launch a variety of attacks on a network. The approach that adversary's use to gather active hosts information on a network segment is to first determine the network and host portion of the address that will be scanned. This can be a simple guess on the subnet boundaries or from intercepted router traffic that specifically provides this information. Either way the adversary then starts with active hosts mapping. This can be done in a variety of ways, of which a popular non-stealthy approach is implemented using a standard ICMP Echo Request "ping sweep" where the host's portion of the address is "pinged" and any response "ICMP Echo Reply" is recorded.

Simply making a connection request to a service to determine if it is available is a problem for the adversary, because this type of port scanning can be easily logged by services listening at the ports. An active port that participates in an incoming connection that contains no data will log this as an error. There exist a number of stealthy scan methods to avoid this error reporting, and still be able to map active hosts and services. They are the SYN, or half-open scan, the Bare ACK scan, and the FIN scan.

The SYN or half-open scan is done by simply initiating a connection request using the TCP synchronization (SYN) request to a destination host address. A SYN/ACK response indicates that there is an available service at the port, and thus a live host. Because the adversary does not continue to negotiate and complete the connection request, the affected host just times out the connection leaving no record of the connection attempt.

The Bare ACK scan takes advantage of the TCP protocol by sending acknowledgements to end host addresses looking for a response. The TCP/IP protocol dictates that all TCP acknowledgements sent to an end host that are not participating in a session with the sender are required to respond with a TCP reset (RST). This reset response is then recorded by the adversary as an active end host.

The FIN scan also takes advantage of the TCP protocol. This scan attempts to close a connection that isn't open. If no service is listening at the target port, the operating system will generate an error message. If a service is listening, the operating system will silently drop the incoming packet. This silent drop is recorded as an active service on an active host.

The previous descriptions of port scanning were associated with TCP ports, which are connection-oriented and therefore give good feedback to the adversary. UDP, or connectionless traffic, responds in a different manner. In order to find UDP ports, the adversary generally sends empty UDP datagrams at the port. If the port is listening, the service will send back an error message or ignore the incoming datagram. If the port is closed, then the operating system sends back an "ICMP Port Unreachable" message

After hosts and services have been identified, the adversary proceeds to identify the Operating System (OS) and its current revision. Operating system identification can also be implemented by misusing the TCP protocol. Because different operating systems respond differently to anomalies in TCP Flag information these responses can then be used to identify each OS. Two techniques that are deployed to map OS responses are called the XMAS scan and the NULL scan.

The XMAS scan is initiated by setting all the TCP flags to logic one. Since this is a non-standard value and is not guided by any documented approach, each OS vendor, if they even take this result into account, can process this response in any way. This unique response to anomalous flag input is use to map the OS. The NULL scan is very similar and sets all the TCP flag bits to logic zero and maps the response, which also helps in identifying the OS.

2.1.1 Dynat Response

Using a Dynat protocol that obfuscates the IP address field will help to prevent mapping and cataloging of active network nodes on any LAN segment. Tools such as NMAP and Nexus rely on building a log file of identified hosts before continuing more specific scans such as Syn, Bare Ack, FIN, and XMAS . With out the ability of a static log that maps active network hosts it becomes very difficult to continue with these more specific types of scanning. Additional types of obfuscation such as using the MAC address fields can add additional complexity to the task of identifying active end host.

It is important to note that running a Dynat protocol on a network does not necessarily guarantee the inability to ever correlate active end hosts to IP address mappings. It just makes it more difficult. In some laboratory tests adversaries with out knowledge of a Dynat protocol being implemented were eventually able to identify the protocol and start the process of correlating active end host to IP address mappings. See *Dynamic Defense ISO experiment 9907 and Attack Strategy for a NAT based Network, DSO National Laboratories.*

2.2 Man-In-The-Middle

The term man-in-the-middle refers to any process or technique that is intended to intercept information between two communicating end nodes and either manipulate this information before forwarding it on to its original destination, or hijacking the session and impersonating one of the originally participating end nodes.

Some forms of man-in-the-middle attack include, Replay attack, TCP hijacking, and ARP spoofing.

A simple form of man-in-the-middle attack is called a replay attack. A replay attack does not necessarily modify information that has been captured, but replays the captured information to gain access to an unauthorized services. For example, capturing a sequence of events between an authorized client and distant server can be replayed at a later date to spoof the distant server into gaining access to its services. To prevent replay attacks strong authentication methods are used that include challenges that are active and don't go stale. But it is important to note that utilizing a Dynat protocol can increase the complexity of gathering the proper information to initiate the replay attack. This complexity comes in the form of possibly needed packet reconstruction to gather all appropriate replay information, and, most importantly, in attempting to identify the servers end point address while it is being actively obfuscated.

TCP hijacking is a more sophisticated attack. It entails the spoofing of TCP packets between a server and client for the purpose of disconnecting the original client with a spoofed alternate. This is accomplished because of the inherent interaction of TCP packets. There are two ways to accomplish this task; hijack a session using the RST flag or hijack a session using the FIN flag.

When using the RST flag attack, packets are sniffed between the original client and server. This allows the attacker to identify IP addresses and associated port numbers. Once this is accomplished, some synchronization is needed to launch the attack. The adversary then waits until a packet from the client is sent to the server with the TCP ACK bit set, in this packet is the Sequence number of the next packet the client is expecting. The adversary now sends a packet with the source address as the server and the destination address as the client with the RST flag set and the expected sequence number from the previous acknowledgment sent from client to server. This causes the original client to disconnect its session with the server and allows the adversary to step in and continue the communication.

Using the FIN flag form of attack is very similar to using the RST flag to disconnect a client. The primary difference when using the FIN flag is that you must set the ACK flag as well when you spoof the packet so the original client receives an acknowledgement for data it just sent. Also in a FIN flag attack the original client will respond with an acknowledgement of the FIN by setting the ACK flag. This assures the adversary that the attack has been successful.

Another form of a Man-in-the-Middle attack occurs at the data link layer of the network and is called an ARP spoof or MAC attack. The ARP spoofing attack sends gratuitous Address Resolution Protocol (ARP) information to targeted hosts to spoof the hosts default gateway (router) making the host believe that the adversaries MAC is the routers MAC. This allows external LAN communications between the targeted host and router to be reviewed. Or locally a target host will be sent a mapping of an IP address and a non-existent MAC of a local server providing the local server with the IP address of the targeted host a non-existing MAC address that will be used by the adversary. This allows for traffic monitoring of desired connections, or even traffic modification of desired connections. This attack is implemented and carried out because of a feature provided when using the gratuitous ARP protocol. This feature allows a host upon boot up, to deliver to all listening hosts and Ethernet switch ports, information that allows these hosts and network devices to update their mapping between an existing IP address and a new MAC address. When a gratuitous ARP is received by hosts and Ethernet switch ports, all previous ARP cache and CAM table entries containing the previous IP-to-MAC address mapping of the host is overwritten by the newly requested association. This allows an adversary with a cleverly written script to issue gratuitous ARPs so new IP-to-MAC mappings will be created with the intent of redirecting traffic flows through an adversary's connected port.

2.2.1 Dynat Response

Dynat protocols can obfuscate multiple parameters associated with the communications protocol stack; see the section *Field Obfuscation* for a list. But none of these protocol obfuscation techniques can explicitly protect a network against the classes of man-in-the-middle attacks previously described.

Standard ways to thwart replay attacks is to include a connection orientated protocol that is able to synchronize the communication of data packets and provide a time stamp element to the

transmissions. Even more protection can be added to also prevent the manipulation of the data by including a digital signature across the header and contents of the packet.

Although Dynat protocols cannot explicitly prevent replay attacks they can increase the complexity of the required attack by reducing the visibility of the target and the time that the target can be legitimately addressed within an address realm.

The TCP hijacking scenario also assumes that a session between two communicating end hosts has been identified prior to implementing the hijacking attack. This information becomes difficult to gather when a Dynat protocol obfuscating the IP address is evoked. Also the TCP hijacking attack revolves around the TCP sequence number identification prior to insertion into the communication session. A Dynat protocol that includes obfuscation of the sequence numbers and/or the IP identification field in a connection orientated session would severely limit the ability of an adversary to successfully hijack a data session.

The ARP spoofing attack, which is basically a session redirection, takes advantage of the MAC layer 2 protocol that is associated with 802.3 (Ethernet) networks. For an ARP attack to be successful the adversary must first record MAC addresses of the intended targets before an ARP spoof can be implemented. A Dynat protocol that obfuscates the MAC address fields forces the adversary to re-identify the intended targets and provide a bogus gratuitous ARP before the address space changes. This process must be continually updated to follow the identified session. This puts a large processing and coordination burden on the adversary.

2.3 Denial of Service

A brief description of some of the popular types of Denial of Service (DOS) and Distributed Denial of Service (DDOS) attacks will be mentioned to formulate a basis to discuss how the Dynat technology may have an impact on preventing or reducing the severity of these forms of attack.

The term Denial of Service (DOS) refers to any activity that can limit, disrupt, or prevent the normal flow of communications between any set of participating end nodes. These Denial of Service attacks come in many varieties, with the most common approach based on high-bandwidth flooding of packets or frames, or on other repetitive activities. Other approaches rely on using standard communications protocols in non-standard ways that disrupt or crash resident operating systems. Some common types of Denial of Service attacks are listed below:

- Syn Flooding
- Ping-of-death
- Smurf
- Fraggle
- Trinoo,
- TFN,
- TFN2K
- Stacheldraht
- MAC

A common denial of service attack is the SYN flood, in which a target machine is flooded with TCP connection requests. The source addresses and source TCP ports of the connection request packets are randomized; the purpose is to force the target host to maintain state information for many connections that will never be completed.

SYN flood attacks are usually noticed because the target host (frequently an HTTP or SMTP server) becomes extremely slow, crashes, or hangs. It's also possible for the traffic returned from the target host to cause trouble on routers; because this return traffic goes to the randomized source addresses of the original packets, it forces the routers into a high rate of route lookups that lacks the local properties of "real" IP traffic, and may overflow route caches.

The Internet Control Message Protocol (ICMP) is used to handle errors and exchange control messages. ICMP can be used to determine if a host is able to respond to an ICMP Echo Request "Ping" on the network. The Ping is sent to a host; if a host receives the Ping packet the host will return an ICMP echo reply packet.

ICMP is used to convey status and error information including notification of network congestion and of other network transport problems. ICMP can also be a valuable tool in diagnosing host or network problems.

The Ping-of-death attack occurs when a large ICMP echo request packet is created that exceeds the specification for the Internet Protocol (IP) which says that a packet may be up to 65,535 ($2^{16} - 1$) bytes in length, including the packet header. But the specifications for most network technologies in use today do not allow packets that big. For example, the maximum Ethernet packet size is 1,500 bytes. To allow large packets to be sent, IP allows the sender to break a large packet up into several smaller packets. Each fragment packet contains an offset value that says where in the larger packet this fragment belongs (i.e. the first fragment will have an offset of zero, the second fragment will have an offset equal to the length of the first fragment, and so on.) This technique makes it possible to combine a valid offset with a suitable fragment size such that (offset + size) is greater than 65,535, the maximum size of a packet.

The problem arises in the way packet fragmentation is implemented by most systems. Typically, they do not attempt to process a packet until all the fragments have been received and an attempt has been made to reassemble them into one large packet. This opens up these systems to the possibility for overflow of 16-bit internal variables, resulting in system crashes, protocol hang, and other problems.

This problem was first discovered in the context of sending ICMP ECHO REQUEST packets, commonly called "ping" packets named after the application program used to send them. Most implementations of "ping" will not allow improperly sized packets to be sent, although there are several exceptions to this (and many systems can be modified to allow it, in any case). Because sending a single, large (65,510 bytes) "ping" packet to many systems will cause them to hang or even crash, this problem was quickly dubbed the "Ping-of-Death."

The Smurf attack takes advantage of the broadcast addresses associated with each network. On IP networks, a packet can be directed to an individual host or broadcast to an entire network. When a packet is sent to an IP broadcast address from a host on the local network, that packet is delivered to all machines on that network. When a packet is sent to that IP broadcast address from a machine outside of the local network, it is broadcast to all machines on the target network if the associated routers allow for this traffic to pass.

IP broadcast addresses are usually network addresses with the host portion of the address having all one bits. For example, the IP broadcast address for the network 10.0.0.0 is 10.255.255.255. Network addresses with all zeros in the host portion, such as 10.0.0.0, can also produce a broadcast response.

The Denial of Service Smurf attack uses ICMP echo request packets and then directs these packets to IP broadcast addresses. These attacks are normally launched from remote locations to generate denial-of-service attacks.

The Ping packet is addressed to a host on a network requesting a Ping reply to the broadcast address of the network. If the host or upstream router does not filter ICMP echo requests to IP broadcast addresses, many of the machines on the network will receive this ICMP echo request packet and send an ICMP echo reply packet back, possibly resulting in network congestion or outages. When the adversary creates the Ping packet, the source IP address is a spoofed source address of the adversary's intended host, normally a server. The result is that when all the machines at the site respond to the ICMP echo requests, they send replies to the spoofed host address. The host is then subjected to network congestion that could potentially make the network unusable.

Attackers have developed automated tools that enable them to send these attacks to multiple associates at the same time, causing all of the associates to direct their responses to the same victim. Attackers have also developed tools to look for network routers that do not filter broadcast traffic and networks where multiple hosts respond. These networks can subsequently be used as intermediaries in attacks.

A similar attack, called "Fraggle," uses directed broadcasts in the same way, but uses UDP echo requests instead of ICMP echo requests. Fraggle usually achieves a smaller amplification factor than Smurf, and is much less popular.

Another form of a Denial of Service attack is the Distributive Denial of Service (DDOS) attack. These attacks can be generically classed as Denial of Service (DOS) attacks with an additional feature that includes compromising many distributed hosts to act as daemons or zombie machines. Each zombie, which is a host machine that has been compromised, carries out a DOS attack resulting in a large distributed and amplified attack..

In order to facilitate DDoS, the attacker needs to compromise a few hundred to several thousand hosts. The process of compromising a host and installing the tool is automated. The process can normally be divided into the following steps

1. Initiate network and then port scans in which a large number of hosts are probed for known vulnerabilities.
2. Compromise the vulnerable hosts to gain access.
3. Install the tool on each host.
4. Use the compromised hosts to launch attacks against networks or other hosts.

The Distributed Denial of Service attacks commonly use the following programs to execute the DDOS; Trinoo, TFN, TFN2K and Stacheldraht.

Trinoo was first found as a binary daemon on a number of compromised Solaris 2.x systems. Malicious code is inserted into operating systems through exploitation of buffer overrun bugs in remote procedure call (RPC) services. The trinoo DDOS attack is created by compromising one of many master systems. These systems are setup with vulnerability scanning tools and root kits to conceal malicious programs, including the insertion of the master and trinoo daemon programs, and a list of vulnerable hosts. DDos attack preparation involves the master(s) scanning for systems exhibiting the vulnerabilities described above (the RPC services) normally on Solaris 2.x and Linux operating systems. A list of vulnerable systems is then passed to an exploit script that compromises each system, sets up and connects a listening shell, and compiles a list of successfully compromised systems.

The list of compromised systems is passed to another script that installs the trino daemon and a root kit via an open tcp port. The DDoS attack begins when the attacker connects to masters via a telnet tcp port and enters a password. Masters then pass command lines to daemons via a UDP port. These commands are password protected and daemons respond to masters on a predefined UDP port. Masters form a list of responding daemons by listening for the text "HELLO" in the data portion of UDP packets originating from the daemons.

Stacheldraht is a distributed denial of service tool, which uses denial of service attacks executed by ICMP flood, SYN flood, UDP flood, and "Smurf" style attacks. The communications between the agents and the stacheldraht master is encrypted and can be automated. Stacheldraht, unlike other distributive denial of service tools, can use TCP connections between the handler and the agents instead of UDP or ICMP.

Tribe Flood Network (TFN), like Trinoo, is a distributed tool used to launch coordinated denial of service attacks from many sources against one or more targets. In addition to being able to generate UDP flood attacks, a TFN network can also generate TCP SYN flood, ICMP echo request flood, and ICMP directed broadcast denial of service attacks. TFN has the capability to generate packets with spoofed source IP addresses. Please see the following CERT Advisories for more information about these types of denial of service attacks.

CA-96.01, TCP SYN Flooding and IP Spoofing Attacks
CA-98.01, "smurf" IP Denial of Service Attacks

A DDOS attack using a TFN network is initiated by instructing a client, or master, program to send attack instructions to a list of TFN servers. The servers or daemons then generate the specified type of denial of service attack against one or more target IP addresses. Source IP addresses and source ports can be randomized, and packet sizes can be altered.

The master communicates with the daemons using ICMP echo reply packets with binary values embedded in the ID field, and any arguments embedded in the data portion of packet. The binary values, which are definable at compile time, represent the various instructions sent between TFN masters and daemons. Communication between clients, handlers and agents use ICMP ECHO and ICMP ECHO REPLY packets.

Like TFN, TFN2K is designed to launch coordinated denial-of-service attacks from many sources against one or more targets simultaneously. It includes features designed specifically to make TFN2K traffic difficult to recognize and filter, to remotely execute commands, to obfuscate the true source of the traffic, to transport TFN2K traffic over multiple transport protocols including UDP, TCP, and ICMP, and features to confuse attempts to locate other nodes in a TFN2K network by sending "decoy" packets. TFN2K is designed to work on various UNIX and UNIX-like systems and Windows NT.

TFN2K obfuscates the true source of attacks by spoofing IP addresses. In networks that use ingress filtering, TFN2K can forge packets that appear to come from local hosts. Like TFN, TFN2K can flood networks by sending large amounts of data to the victim machine. Unlike TFN, TFN2K includes attacks designed to crash or create instabilities in systems by sending malformed or invalid packets.

Another form of denial of service attack located at the MAC layer of the OSI model and implemented in a brute force way is referred to as the "MAC attack". The MAC attack takes its name from the acronym of the original world Media Access Control (MAC). This attack takes

advantage of the memory needed to store the MAC address to port mappings within Ethernet switches.

When Ethernet switching is used to provide network communications, the switch builds a content addressable memory (CAM) table that contains a mapping of the source MAC address of an Ethernet frame and its associated port. This is needed to allow the switch to determine the destination port of a transmitted Ethernet frame. All workstations and servers on a local segment have a unique MAC address, which is associated by the Ethernet switch with its interconnected port. This mapping allows the switch to direct the Ethernet frames to their proper destination. The information to build the CAM tables is retrieved by the switch with a process called Address Resolution Protocol (ARP).

An adversary that wants to deplete the available memory space creates a script that sends out a large volume of gratuitous ARP's, which are stored in the CAM tables of the Ethernet switch. Because the volume of these ARP's is greater than the designed capacity of the Ethernet switch it causes the switch to reset or stop forwarding Ethernet Frames, thus creating a Denial of Service situation.

2.3.1 Dynat Response

For a TCP SYN attack to be successful, the IP address of the target must be known. If a Dynat protocol is being deployed to obfuscate the IP address space of a network, then the Syn flood attack can be thwarted. The degree to which this type of attack can be thwarted depends on the adversary's desire and available resources. But the hacker who attempts to gain information about the target can be further thwarted if the Dynat protocol also obfuscates TCP or UDP port numbers.

The Ping-of-death attack also relies on the need to address the Ping packet to an intended target. If the target address cannot be easily identified then the adversary may pursue more easily identified targets. Also because the Ping-of-death packet relies on IP fragmentation, it is quite possible that only some of the fragmented packets will reach their intended victim before the Dynat protocol changes the IP address realm.

The Smurf and Fraggle attacks take advantage of the broadcast addresses associated with each network. On IP networks, a packet can be directed to an individual host, or broadcast to an entire network. When a packet is sent to an IP broadcast address from a host on the local network, that packet is delivered to all machines on that network.

The difficulties in launching this attack when a Dynat protocol is obfuscating the IP address space is finding a valid IP address to use as the source IP address. An adversary that wants to launch this attack against a machine hosting a service must first determine the IP address of the server. This becomes a difficult task when a Dynat protocol with IP and MAC obfuscation is utilized.

Also if this attack is being launched externally from the intended target network, proper ICMP packet filtering located on the inter-connecting Router can prevent this packet from ever reaching its intended target.

It must be noted that Dynat protocols can obfuscate the entire IP host address space of a network, but must also recognize and respond to all default addressable broadcast addresses, normally represented as all ones or all zeros in the host ID portion of the frame or packet.

For all the previously reviewed Distributed Denial of Service attacks that included the following: Trinoo, TFN, TFN2K and Stacheldraht, the best defense against DDOS attacks is to prevent

initial system compromises. Generally, this involves installing patches, anti virus software, using a firewall and monitoring for intruders. However, an active Dynat protocol can add an additional layer of defense when it comes to the network and port scans that are used to find the vulnerabilities that normally precede these types of attacks. Uncorrelated data from scanning tools that are attempting to map networks and services on a Dynat enabled network can frustrate and thus help thwart compromise attempts. See *IA 0106 Experiment, 27 March 2000 BBN*.

The MAC attack uses the gratuitous ARP to pass on MAC address-to-port mappings to the attached Ethernet switch and all participating end hosts. This attack, with the implementation of a Dynat protocol using MAC obfuscation, can be detected by participating hosts and ignored. Unfortunately, it's the Ethernet switch that is the target of this attack. Without the ability of the Switch to determine valid MAC addresses from bogus ones it remains susceptible to this type of attack.

The previously discussed MAC lock feature available in some Ethernet switches allows broadcast addresses to originate from ports and thus would not prevent this type of attack.

3. Traffic Analysis

Traffic analysis is the process of monitoring the nature and behavior of traffic, rather than its content [20]. It is concerned with where they come from, where they go to, how long they are, when they are sent, how frequent or infrequent they are, whether they coincide with outside events like meeting, and so on, thereby serving as an effective tool to infer sensitive information about the applications and underlying systems. It can also extract important information for intrusion and cryptanalysis attacks [19].

By changing fields on packet headers, regularity in traffic may be altered so that an observer would not be able to derive any useful information from network traffic. We term this hiding of information to confuse adversary information obfuscation. Changing server endpoint may also potentially remove location information from network traffic to achieve anonymity. This report evaluates traffic analysis resistance of Dynat-based information systems.

In section 3.2, a brief description on traffic analysis techniques is given, along with identify a generic protection model for resisting traffic analysis, and show how existing techniques fit into the model. In section 3.3 an adversary model upon which this analysis is based is described. An analysis of traffic analysis resistance of Dynat based systems is presented in section 4. The analysis examines the resistance provided by a generic Dynat system on two aspects: information obfuscation and route hiding.

3.1 Hindering Traffic Analysis

3.1.1 Types of Traffic Analysis

Many traffic analysis techniques exists. The following describes some traffic analysis scenarios:

3.1.2 Network activity identification

These techniques attempts to identify “Who is communication with whom?” “When and how often do they communicate?” and “Is there any relation among communications?”

Communicating entities and their location is normally printed on the packet headers. A number of traffic statistics can be collected to describe their activities, such as number of endpoints a host

connects to, connection timestamp and duration, packet/connection frequency, message volume, connection type (e.g. transport service, protection service, network forwarding privilege), etc.

Such information may be used to plot dependency map of the target network, identify critical hosts, trust/ access relationships and network status, correlating connections among different hosts to identify workflow (e.g. distribution of document through email attachment), correlating workflow to real world events and so on.

3.1.2 Network Reconnaissance

The attack type attempts to gather network level information like topology, routing information, LAN technology, accessibility through firewall, number of hosts in each subnet, number of service on a host may be exposed in network traffic, their address, the use of security protection services. For examples,

- Servers on a network can be identified by noting the source endpoint of syn-ack packets (packets with TCP flag syn and ack bit set);
- Number of hosts in the LAN segment can be obtained using ARP response packets;
- MAC address can be used to infer type of interface, e.g. Cisco interfaces;
- Gateway can be identified by many-to-one IP-to-MAC address mapping in network packet headers;
- Distance (number of hop) between different subnet can be deduced by noting the difference in TTL value to the default TTL;
- Accessibility through firewall can be identified by observing endpoints of packets passing through it;
- Use of IPSec ESP and its scope can be discovered by noting IP protocol and endpoints; etc.

3.1.3 Application Fingerprinting.

OS/ Application fingerprinting attempts to answer questions like “What application is used?” and “What type of message is sent?” using traffic information.

Applications emit different traffic signatures when performing different activities. The signature can be found in packet header fields (like IP protocol, default port, TOS, window size, TTL, flag), packet timing information (like packet rate, packet inter-arrival time, packet timestamp), message volume, and message exchange pattern (how many messages are exchanged in a connection, size and timing of each message.)

For instance, OS versions may be uncovered with unique combination of DF flag, TTL, window size and TOS [21]. Type of applications, such as traffic real-time (e.g. voice or video applications), burst type (e.g. FTP), session-based (e.g. Telnet, SSH) applications, can be easily differentiated by the traffic pattern, packet periodicity and transport layer protocol (UDP or TCP), among others, of the traffic. Message exchange pattern, as indicated in how data is exchanged in term of number of message exchanged in single burst, their size and packet timing information,

can also help in fingerprinting an application and its messages. Unique packet size can also indicate a particular application or narrow the search.

Another example is anonymous web browsing via third-party proxies. URL information of the website visited and response HTTP data are usually hidden from observer by encryption. By counting the number of files downloaded (each in a separate TCP/IP connection) and their size, and matching the fingerprint to a website signature database, attackers can potentially identify the websites visited [20]

3.2. Traffic Analysis Protection Model

Traffic analysis attacks can be *prevented* from any third party adversary if

Condition 1: No information can be obtained from traffic by observing over an infinite duration at any link, and

Condition 2: Route taken to deliver packet is hidden.

The first condition focuses *traffic understanding* based on information observed at any static point in the network. Such traffic information as packet header values, packet size, volume and timing information may be used to infer other information to improve understanding on network activities. The second one covers the fact that a persistent adversary can attempt *packet tracing*, which is to trace packet across network elements to identify communicating entities.

3.2.1 Information obfuscation

There is two ways to provide information obfuscation:

- By blending/ camouflaging protected traffic into background traffic;
- By obfuscating all network traffic and hiding protected traffic within.

In the first case, protected traffic would be transformed to assimilate normal traffic so that it is totally or mostly indistinguishable from normal background traffic. If background traffic is large, it may be difficult for attackers to identify the protected traffic. The main limitation of this method is that any addition of protection services to the protected traffic can be easily detected due to its distinction. Moreover, this would require background traffic modeling, and large background traffic. Traffic padding (i.e. injecting dummy packets) is required if normal traffic is light. This case is relevant in sending traffic over public networks.

The second case could offer better security. By applying obfuscation consistently to all traffic, this could ensure that the protected traffic hides well in normal traffic. Level of obfuscation and security can be easily controlled. Traffic padding may be necessary if background traffic is light. To remove any distinguishing information, the key to obfuscation is consistency: either consistently randomized or consistently conforming to a particular pattern. This may be the case in LAN environment if there is total control over a network.

Traffic information can be categorized into *packet header fields*, *packet size*, *packet timing* and *message volume*. Providing obfuscation to these information types would hinder traffic analysis. For the second can be obtained from traffic by observing over an infinite duration at any link, can be provisioned if

- All packet headers are randomized,

- Packet size is randomized or made consistent.
- Inter-packet arrival time is randomized or made consistent.
- Total message rate on any link is randomized or made consistent.

Dummy packets would be injected to facilitate packet timing and message volume obfuscation.

3.2.2 Route Hiding

Providing untraceability has been well studied as an important component in achieving anonymity, and in turn traffic analysis resistance. Proper implementation removes ability of any attackers, active, passive, insider or multiple of them, to identify communicating entities by packet tracing.

Untraceable communication is frequently approximated using *mixing* and *rerouting* [10]. Data is routed through intermediate nodes, call *mixes*, which may reorder, delay and pad traffic (fix message size and dummy message injection) to complicate traffic analysis by removing correlation between incoming and outgoing packets. A single perfect mix adequately complicates traffic analysis, but a sequence of multiple mixes is typically used because real mixes are not ideal. A number of mix-based techniques have been suggested to address different anonymity requirement and threat model. They include onion routing [18][22] and crowd [15].

Coincidentally, mixing and rerouting employ similar obfuscation types as identified in section 3.2.1, as the same traffic parameters that provide useful information to understand traffic also allow packet tracing across mixes.

Effectiveness of various route-hiding techniques can be expressed in their vulnerability to various attacks such as message coding, timing, message volume, intersection, flooding and collusion attacks [9]. Crowd argues that packet tracing can be prevented if re-routing is performed over different administrative domain.

The research on route hiding focuses predominantly in WAN. In LAN, untraceability can be simply provided with the use of broadcast traffic. Of course, to achieve anonymity in both WAN and LAN cases, packet header and payload has to be sanitized to remove location and identity information.

3.2.3 Related Works

Traffic analysis can be prevented by presenting intruders a neutral traffic pattern. That is, each host sends every other hosts in the network the same volume of messages. This goal can be achieved with the use of dummy packets when regular traffic is too low, rerouting packets via one or more intermediate nodes and delaying packets. It is assumed that encryption is performed at transport layer [14]. The basic methodology is adopted in NetCamo [11], adapting for real-time communications.

A related problem to traffic analysis resistance is anonymity. Anonymity is concerned with secrecy of sender, recipient, and their relation from network eavesdropper [16]. Anonymity shares suspiciousness among participating entities.

Providing anonymity has been recognized as a means to resist traffic analysis [17][18]. David Chaum described "traffic analysis problem" as the problem of keeping confidential who converses with whom, and when they converse [10]. The network operator or an intruder could easily observe when, how much and with whom the users communicate (traffic analysis) [16]. The reason for these definitions is probably due to the concern in privacy over public network.

Anonymizing may not be sufficient. Simple anonymizing technique only performs address field obfuscation, in conjunction with encryption or address/ identity sanitation of packet payload. This is clearly insufficient in resisting traffic analysis, as other packet header fields are not obfuscated, nor packet size, packet timing and message volume are hidden.

The absent of identity and location information would not stop a persistent attacker. Anonymized session may still be vulnerable to fingerprinting to narrow target session to attack, which may then subject to session hijack or cryptanalysis. As noted in section 3.1.3, an adversary can identify that the websites visited using Safeweb anonymizing tool by fingerprinting, even if HTTP data in request and response packet is encrypted or sanitized.

Traffic analysis may also obtain some useful information in event where anonymity is insufficient or difficult to deploy. For instance, if anonymity is only shared among sensitive organizations, an increase in network activity may be correlated to heightened alertness.

Apparently, anonymizing alone is not sufficient; much information is contained in network traffic. That is, *anonymity is necessary but not sufficient for resisting traffic analysis*.

Nevertheless, some advance mix-based anonymizing techniques does provide a good degree of traffic analysis resistance. In addition to address field obfuscation, these anonymizing techniques provide route hiding. As mentioned in section 3.2.2, they use some forms of information obfuscation like packet header fields, packet size and timing and message volume obfuscation. As such, some advance anonymizing techniques can potentially provide enough information obfuscation and route hiding to *prevent* traffic analysis.

The requirements to hinder traffic analysis has been discussed in section 3.2, therefore the focus will now shift to discuss the degree of traffic analysis resistance provided by some existing techniques. Table 1 summarizes mechanisms provided in various techniques for resisting traffic analysis. Anonymizer [8], crowd [15] and onion routing [18][22] are examples of anonymizing techniques, and NetCamo [11] protects against traffic analysis. Dynat is also included for comparison, and its analysis would be given in section 3.4.

Table 1 Comparison of Techniques

Techniques	Information obfuscation				Route hiding
	Header fields	Packet Size	Packet Timing	Message volume	
Anonymizer	Initiator addresses hidden in rerouted packets.	No.	No.	No.	No.
Crowd	Packet encrypted in rerouted packets.	No.	No.	No.	Limited.
Onion Routing	Packet encrypted in rerouted packets.	Yes.	Yes.	Yes, but optional.	Yes.
NetCamo	Initiator and responder addresses encrypted in rerouted packets.	Yes.	Optional. May send immediately.	Yes.	Yes.
Dynat	Yes, but subjected to constraints.	No.	No.	No.	Within LAN segment only.

- Anonymizer allows hiding of initiator IP address to the responder by acting as proxy. Traffic from initiator to proxy may be encrypted using SSL. Although packets are rerouted at the proxy, route hiding is minimal as packet tracing can be simply performed by correlating inbound and outbound traffic at the anonymizing proxy.
- Crowd hides route over multiple anonymizing nodes, forwarding to a next random anonymizing node based on a probability. Packets are encrypted in each hop except forwarding to actual responder. As packet size, timing and message volume is not obfuscated, route hiding is limited and fingerprinting is possible.
- In onion routing, initiator encrypts each packet multiple time, one for each anonymizing node. Packet size, timing and message volume are obfuscated. Route information is reasonably protected from message coding, timing, message volume, flooding and collusion attacks
- NetCamo attempts to remove traffic pattern by maintaining fix packet size and traffic rate between two communicating entities. Packet is rerouted via multiple nodes, with initiator and responder IP address encapsulated and encrypted as part of payload. Note that both onion routing and NetCamo perform all mechanisms required for resisting traffic analysis. The difference lie on their emphasis: onion routing on anonymity and NetCamo on traffic analysis resistance. In onion routing, initiator encrypts multiple times to protect against compromised anonymizing nodes, mixing (reorder, delay and pad) to hinder packet tracing; NetCamo aims to manipulate traffic pattern to hide traffic information by traffic padding.

3.3 Adversary Model

Types of adversaries are defined as follows.

- External attacker: attacker at IP network beyond organization perimeter, typically between routers/ gateways.
- Internal attacker: attacker at switched or broadcast LAN segment. An internal network may be segmented using router. Broadcast LAN protocol in a LAN segment is not carried cross segments.
- Active attacker performs active intervention like generating traffic, modifying traffic in transit in term of packet header value, packet timing and packet size.
- Passive attacker only monitors and collects traffic for analysis.
- Insider refers to attackers at compromised machines that host communicating processes, or compromise system network element (Dynat gateway in our case) where traffic-to-analyze traverses.

Both internal and external attackers scenarios are considered. That is, attackers may be located at external, internal network or both, and are able to correlate data collected at multiple points along the packet-traversing path.

There are two aspects for analysis of traffic analysis resistance, namely information obfuscation and route hiding. In the study of information obfuscation, we only consider the case of passive attackers. The case for active attackers is examined in a separate study on adversarial white-boarding.

In the study of route hiding, we assume both active and passive attacks. Active attackers can modify network traffic, by changing packet header values or enforcing certain timing and packet size signature on traffic, to facilitate packet tracing. Passive attackers correlate based on existing traffic signatures.

Insiders are not consider, i.e. compromised Dynat gateways or machines where communicating entities are hosted. Both communicating entities are assumed trustful, and each side should administratively own Dynat gateway, if any. Both parties should therefore have vested interest in securing hosts and gateways.

3.4 Security Assessment

In this section, an attempt is made to qualify the traffic analysis resistance of Dynamic NAT systems: its usefulness, limitations and recommendations for improvement.

Two important characteristics of Dynat that hinder traffic analysis are *information obfuscation* and *route hiding*. Information obfuscation is fundamental in Dynat as traffic delivery parameters are changed in a manner unknown to network observers. The degree of obfuscation affects how much the network observer can understand the intercepted traffic. Routing hiding is one of the important components in providing anonymity, one research focus in protecting against traffic

analysis. The following describes how route hiding may be provided in Dynat systems. A comparison with other techniques is shown in Table 1.

3.4.1 Information Obfuscation

Dynamic NAT systems work on the principle of obfuscating packet header fields. Obfuscation distorts the information carried in the packet headers, and thus would potentially degrade the accuracy of traffic analysis. As mentioned previously, only traffic analysis is consider based on traffic information. Information in packet payload should be hidden from network traffic observer using techniques such as encryption.

Dynat provides obfuscation to packet headers. Generally, the more packet header fields that are obfuscated, the more difficult it is for an adversary to analyze the traffic. Fields that are commonly obfuscated include source and destination IP addresses, UDP/TCP port and MAC addresses, IP identification and TCP sequence number. Other packet header fields can also be potential targets for obfuscation.

Endpoint Obfuscation

Endpoint obfuscation removes association of endpoint to actual communicating entities. This can be achieved by changing addresses and port fields in Ethernet II, IP, UDP/TCP packet header, etc. This provides a certain degree of anonymity on packets to communicating party. The effect is that a communicating entity would assume different addresses, and this would result in inaccurate analysis, such as number of hosts in a subnet, from simply observing network traffic.

If address values available for Dynat process is exclusive, that is only the process can rightfully assign those address values, then relationship of endpoint to communicating entity is many-to-one. Correlation of various address values to a single communicating entity would be possible after observing network traffic for a long time. On the other hand, it would be more difficult for traffic observer if many Dynat processes reuse the same endpoint. In this case endpoint would give little indication to the communicating entity, as relationship of endpoint to communicating entity becomes many-to-many. One possible method is to time-synchronize Dynat processes so that different processes use an endpoint at different time. The overlapping of address space for obfuscation is inherent for multiple processes hosted on the same machine, and among all hosts behind a Dynat gateway from an external observer.

The larger number of address available for hopping, the greater is the obfuscation in term of accuracy of traffic analysis. However, size of address space would be irrelevant if endpoint correlation, i.e. correlating packets to a single endpoint, can be performed based on other traffic information. For instance, incremental IP identification field in different packets is good indication that the packets originate from same host. Endpoint correlation can also be approximated if they are running the same OS and applications, identifiable with unique fingerprints as discussed in section 3.1.3, or sending over certain links.

Session Obfuscation

Dynat can also achieve a certain degree of *session obfuscation*, where the session a packet belongs to can be hidden from network traffic observer. Session represents basic unit of interactions among communicating entities. In TCP, session denotes a connection. In UDP, a

connectionless protocol, a session demotes an exchange of data in action-response communication. They are useful in identifying communication pattern, fingerprinting and data collection. Fields to obfuscate to achieve this include endpoints and other fields that can be used for correlation such as identification and flag in IP packet header, sequence and acknowledgment number, window and flags in TCP packet header, and Security Parameter Index (SPI) and sequence number in IPSec header. Although IP header fields has more relevance to a host than to a connection, weak correlation still exists and this can be reinforced with the use of other information such as timing (packets that appear in short time) and location where the packet is observed.

One additional benefit of changing addresses is that route taken to deliver the packet may change as a result. The best case is for the routes for each destination address to differ totally from the sender to recipient. As such, an observer needs to intercept traffic at all routes to correlate the session. In reality, however, route taken often shares common physical links, like in the case of IP network where no source routing is used.

The rate of Dynat encoding change affects session obfuscation. More encoding changes within lifetime of a session creates more confusion to the observer collecting session level information and therefore is more effective against traffic analysis. Dynat encoding change can be perceived as decomposing a logical session into a number of sub-sessions. This disrupts session level traffic information gathering such as timing signature or message volume transferred in a connection. This is especially effective in the case of packet or frame based process control where encoding change occurs at every packet or frame. One disadvantage to high encoding change rate is, in the case where anonymity is absent or minimal (e.g. a local eavesdropper on a direct link to a host), an observer may monitor the changing of fields to reverse-engineer keying information, given known algorithm or the possession of legitimate hardware/ software.

Other fields can be obfuscated as well. The effect of field obfuscation on resistance of traffic analysis is dependent on traffic analysis techniques and traffic information they rely on, and the fields being obfuscated.

Field Obfuscation Constraints

Field obfuscation has some constraints. One requirement is to properly deliver the packet to Dynat process at destined host for recovery. The requirement would affect field obfuscation in LAN and network layer. For instance,

- Destination IP address assigned has to be routable if a packet is sent over IP network;
- TOS should not be changed to maintain a certain quality of service;
- Value of TTL should be at least the network diameter;
- Changing MAC address in switched LAN segment may fail to forward to the correct link where the destined host is located and would increase memory requirement for switch's forwarding table; etc.
- IP header checksum should not be changed to avoid packet dropping at intermediate routers.

Hiding into Background Traffic

We have briefly discussed in section 3.2.1 on how traffic may be obfuscated to hide within background traffic. Sending randomly obfuscated traffic over normal background traffic would be apparent. This would be the case of sending dynat-ed traffic over public network. To blend into background traffic, traffic modeling would be required to obfuscate to assimilate a class of background traffic. For instance, in public network traffic with large HTTP traffic and little encrypted traffic, it may be stealthy to masquerade as HTTP traffic, and to sanitize fields and application data to remove sensitive information. However, there is a limitation on obfuscating IP address while keeping it routable. There is little anonymity here.

IPv6 offers a better environment for hiding traffic. There is a wider range of IP addresses and all packets are encrypted at transport layer. This may offer more flexibility in providing better anonymity, and data and traffic secrecy. In contrast, sending obfuscated traffic in LAN environment over similarly obfuscated background traffic would provide much better traffic hiding. Amount of background traffic may affect effectiveness of traffic analysis. If background traffic is heavy, the signal-to-noise ratio is very low. Adversary would require high computational power to sift through large amount of traffic, hence making traffic analysis more difficult. However, most LAN segment is lightly loaded most of the time. Dynat provides no feature to inject dummy packets.

Effects on Traffic Analysis Techniques

Various traffic analysis scenarios have been discussed in section 3.2.1. The following information highlights the effect of information obfuscation in greater detail.

1. Network activity identification

By obfuscating endpoint information, thereby removing association of the endpoint information to actual communicating entity, many of the attacks in this category will be adversely affected to give inaccurate result. For instance, number of communicating entities is unclear from the inspection of endpoint alone. However, hiding endpoint information is not sufficient. For instance if IP identification number is not obfuscated for instance, different packets with obfuscated endpoints may be correlated into the same host, and therefore likely to originate from or destined for a single communicating entity.

The obfuscation of session information would disrupt the collection of session-level statistics and data such as connection count and amount of data transferred in a session, and can resist cases like the above-mentioned session correlation using IP identification field.

However, field obfuscation alone is insufficient to hide session. There are some constraints on field obfuscation, and the connection a packet belongs to can also be derived by correlating on message volume or timing information, especially in light background traffic.

2. Network Reconnaissance

Obfuscation in Dynat focuses on hiding LAN protocol, IP and UDP/ TCP packet headers. This would be quite effective in many attacks in this category mentioned in section 2.1.2. For instance,

attackers cannot identify servers using syn-ack packets if endpoints and TCP syn-ack flag are obfuscated. Also, attackers cannot identify gateway by mapping MAC address to IP address if they are obfuscated.

ARP is an example of network information (source and destination IP and MAC addresses in this case) being revealed in packet payload. More information is printed on packet payload of network protocols such as routing protocols and CDP. Payload encryption has to be provided to hinder network reconnaissance. This will be discussed in section 3.4.

3. Fingerprinting.

In the example of anonymous web browsing, by changing the endpoints and TCP flag, attackers may be unable to identify the correct number of connections and the bytes transmitted in the connections, thereby failing to match the fingerprint to the correct website signature correctly. Field obfuscation can also modify packet header fields (DF flag, TTL, window size and TOS) to avoid OS identification and even mislead adversary.

Apparently, field obfuscation can help to hinder some attacks. However, Dynat does not obfuscate important information such as timing and message volume. It is possible to correlate packets to session (using unhidden fields and timing information, etc), and subject the session to attacks such as fingerprinting and crypt-analysis. Unique packet size may also reveal the type of application.

3.4.2 Route Hiding

Anonymity can be provided with endpoint obfuscation and route hiding. By obfuscating endpoint fields such as IP addresses, MAC addresses and TCP/UDP port, Dynat process could remove, to a certain degree, correlation of endpoints obtained in network packets to the end hosts or processes. This section attempts to provide an analysis to the degree of route hiding that may be provided by Dynat.

For the purpose of this study, further categorization of the types of topologies identified in [16] Can be accomplished according to the position of Dynat process.

- “Dynat at gateway” occurs in servers at LSDS and all hosts in Local Router to Local Router (LRLR) and Gateway-to-Gateway (GWGW), where Dynat exists between local router/ gateway and remote router/ gateway or hosts, and Dynat is not used in LAN segment.
- “Dynat at end hosts” denotes the scenario where Dynat is used among dynat-ed hosts in LAN segment. This includes the cases for clients in LAN Segment Distant Servers (LSDS) and all hosts in LAN Segment Local Server (LSLS).
- “Dynat at end hosts and gateway” is the case for Segment-to-Segment (SS), where Dynat exists between local router/ gateway and remote router/ gateway or hosts, and in LAN segment.

Dynat at Gateway

There is no route hiding for Dynat gateway. As there is no re-routing, each packet would be traceable to the gateway. Traffic pattern between gateways are no hidden either.

For the case of external observer, packet is not traceable beyond the gateway. However under multiple-adversary threat model with observers in both internal and external network, packets are traceable across the gateway by correlating packets across the gateway on timing, message volume or other unhidden fields (if field obfuscation is inadequately performed). That is, a packet at the external interface of the gateway can be correlated to match the corresponding one at its internal interface. Without field obfuscation the packets in LAN segment, the correlation would reveal the packet's real source and destination endpoint in its header.

Dynat at End Hosts

Proper field obfuscation may remove any correlation on the packet header to sender or recipient. Anonymity requires one additional component: route-hiding.

Broadcast contention-based LAN segment, like that of hub-based Ethernet LAN, offers the best route-hiding. Such LAN segment ensures that packets are sent to all hosts in the LAN, providing no traces to the recipient. With proper obfuscation, no observer on any link, either external or internal, or single or multiple adversaries, can easily identify sender and receiver within the LAN segment, except with precise timing analysis over multiple internal observers.

The larger number of participating end-processes in a broadcast LAN segment, the better is the anonymity. Segmenting a network into LAN with smaller group using routers would degrade anonymity. MAC address is irrelevant here to send packets within the broadcast LAN segment, and can therefore be set to some fictitious values or obfuscated. However, one performance implication of broadcast LAN is scalability. Broadcast LAN may also suffer from flooding DoS attacks.

In contrast, switched (in unicast case) and routed internal network provide less anonymity, as presence of packets on links indicates relative location of end hosts. One obvious case is that a packet observed on local link to a host in a switched network indicates the host as either the sender or recipient. Packets observed on a routed link to a LAN segment with few hosts also offer little anonymity. This environment is also vulnerable to multiple adversaries. Multiple observers at different links in the switched or routed network can narrow the possibility or even trace to the end host by matching header fields such as MAC address, or by timing and message volume. There are also constraints on the range of values for obfuscation. The exceptions to this argument are the uses of broadcast and multicast (using switches without multicast support) addresses in switched network, which is similar in behavior to the case in broadcast LAN.

To external observers, on the other hand, anonymity can be maintained among communicating entities in the LAN segment by removing correlation with proper field obfuscation.

Dynat at End Hosts and Gateway

There is no anonymity for gateway, as in the case of "Dynat at Gateway", as packets are traceable to the gateway.

Dynat processes at internal and external interfaces of the gateway perform field obfuscation from one form to another, hence avoiding passive and active message coding attack (tracing across gateway by identifying/ marking packets with special marker). However, there is no additional protection with the use of Dynat between local gateway and remote processes. This is due to the fact that multiple adversaries can use packet size, timing and message volume information for packet tracing across gateway. The degree of anonymity on communicating entities behind gateway is identical to the case of “Dynat at end hosts”.

3.4.3 Limitations and Recommendations

Dynat has no provision to obfuscate packet size, timing and message volume. We have indicated that they should be hidden from observer to achieve information obfuscation in section 2.2.1. We have also illustrated that the information would be useful for some traffic analysis techniques in analyzing network activities and fingerprinting.

It is possible that timing and message volume of a connection can be distorted by field obfuscation to a certain degree, that the packets might not be perceived to belong to the same connection. However, their correlation to a connection may be apparent in light background traffic (which is not unusual especially in LAN environment). For instance, with few communicating processes that communicate infrequently, any burst of packets would most probably belong to a single session, regardless of whether the packet headers are obfuscated. We have noted too that sending obfuscated traffic over public network is easily observable among normal unhidden background traffic, therefore vulnerable to endpoint and session correlation. Obfuscated packets can be collected for fingerprinting, cryptanalysis or other forms of attack. Moreover, timing and message volume information are also known to be used in packet tracing attack to compromise anonymity [9], relevant across Dynat gateway in our case.

Dynat’s resistance to traffic analysis can be further improved with ability to

- . Inject dummy packets,
- . Vary timing (by varying packet transmission delay) and
- . Pad packet to a certain size.

These measures are useful for both information obfuscation and route hiding. They would change traffic characteristic, such as unique packet size, message volume transferred and timing signature, of application to avoid fingerprinting or other attacks, especially in light or normal background traffic. Injecting dummies would also help to reduce signal-to-noise ratio. For example, communicating pattern would be hidden over switched or routed network if dummy packets were sent to hosts other than the rightful recipient. Dummies also increase computational burden of traffic analyzer.

Anonymity can be improved by applying these measures between Dynat gateways, as colluding internal and external observers would be more difficult to trace packets across the gateway, therefore improving the case for “segment to segment” topology. It would also stop active attackers using timing attack (by enforcing a particular timing signature on a connection) for tracing packets across gateway.

It has been noted the weakness of Dynat in providing anonymity to Dynat gateway from external network. This can be enhanced with incorporating of mixing and rerouting, as offered in techniques such as onion routing [18][22], crowds [15] and NetCamo [11].

3.4.4 Other Issues

Data Confidentiality

In most cases, data confidentiality is necessary to maintain traffic analysis resistance of Dynat systems. As discussed, the resistance is provisioned primarily on information obfuscation. Some applications may leak information in the network packet and unwittingly jeopardize any Dynat resistance.

For instance, there is little need for an adversarial observer to fingerprint a server using traffic information in network packets if the application type and its version are printed on its packet payload (e.g. server information in HTTP). Reading SMTP packets to an email server could easily identify the communication pattern. IP address printed on packet payload also eliminates anonymity. Plenty of information is also available in network and OS-related protocols. For example, routing information in routing protocol (BGP, OSPF, etc.), IP address, capability (e.g. switching, routing) and application version in Cisco Discovery Protocol (CDP), and OS version, services available, hostname in Microsoft Windows Browser Protocol.

It should be emphasized that data confidentiality is very important, without which traffic analysis resistance has little significance. Since the topic of traffic analysis focuses primarily on the information obtained in the traffic rather than the content, the analysis will not consider information carried in packet payload. Therefore no assumption on the availability of data confidentiality protection is made.

Implications of Framework Design

Some header fields could be introduced with Dynat protocol that can become new target for traffic analysis. These header fields may be necessary for coordinating the communicating Dynat processes, such as session identifier, anti-replay sequence parameters. Handshaking messages can also become targets for traffic analysis.

For instance in the case of data security using IPsec ESP, it is noteworthy that IPsec ESP provides network layer encryption that hides some traffic information useful for traffic analysis. With IPsec ESP in transport mode, transport layer protocol and above are encrypted. IPsec ESP in tunnel mode encrypts the entire IP packet and encapsulates within a new IP header. That leaves less information for traffic analysis. However, additional protocol header produces new header information that could be useful to adversarial observer. For instance, each unique SPI between two IPsec gateways may denote a communicating pair behind the gateways in a direction. This parameter has to be obfuscated in order to hide each end-to-end session within IPsec gateway-to-gateway traffic. Sequence number, that has relevance to a session, also requires obfuscation.

Clearly, the additional packet headers and handshaking messages are highly dependent on specific Dynat implementation, it is therefore important that their effect on traffic analysis resistance are studied.

Examined resistance of Dynat to traffic analysis has been conducted on two properties: information obfuscation and route hiding. From the analysis, it has been shown that Dynat could provide a certain level of resistance from revealing traffic information to illegitimate observer. Unlike techniques like onion routing and NetCamo, Dynat could be efficiently and effectively deployed in LAN environment. The following is a summarization of the findings on improving resistance of Dynat in the following.

1. The more packet header fields being obfuscated, the better is information obfuscation.
2. The higher the Dynat encoding change rate, the better is information obfuscation.
3. Anonymity is better on broadcast LAN segment, and the more hosts the better to disperse the suspiciousness.
4. Sharing of same endpoint address space among multiple hosts provides better resistance than the exclusive case.

It has also been recommended that providing additional information obfuscation by traffic padding, varying packet transmission delay and padding packet to fix size. Incorporating mixing and rerouting can also improve anonymity to Dynat gateway.

It should be noted that data encryption is vital to prevent the leaking of information in packet payload, to complement traffic analysis resistance provided by Dynat. Furthermore, Dynat traffic analysis resistance does not protect against insiders, and thus authentication and authorization mechanism has to be employed to defend against insider attacks.

The analysis focuses primarily on generic Dynat systems instead of any specific implementation, and thus no consideration is given for control messages for operation such as key management and system control. Nevertheless, the resistance on the control channel is just as important.

Summary

As this report points out, the deployment and use of Dynat technology is dependent on many different factors. These factors can influence the overall effectiveness of the particular Dynat type chosen. The type of Dynat is identified by, and dependent on, its associated attributes and their implementations. It is critical to understand the network environment and the Dynat implementation to determine how this interaction may be used to enhance the security of a network and its associated services.

No security technology that is added to aid the protection of a network, process, or device is inserted without some measurable form of hindrance or degradation. This hindrance can manifest itself in multiple ways. With the Dynat protocol this hindrance or degradation can be shown in the network protocol processes, in some applications that are supported across the network, and the security devices and their implementations found in modern networks. The amount, and severity of, the degradation is dependent on the type of Dynat implementation. As the techniques that are being used vary, so will the difference in the interaction between applications and underlying network protocols.

An area that shows promise for security enhancement utilizing a Dynat protocol is in the area of Intrusion Detection Systems (IDS). If a network based Intrusion Detection System can be made aware of the Dynat protocol, it will be able to detect anomalous activity. Extensions implemented within Dynat can allow for state information to be kept for packets that are detected operating outside the current active address realm. This state information can be used to create filters that filter out packets outside the current address realm. This information can then be used to quickly identify adversarial packets injected into a protected network.

Dynat can have a role in network security, but it is not a silver bullet when securing a network. Its improper use and implementation can have a detrimental affect on the network and services it is deployed to protect. Understanding the underlying Dynat process and how this process may help or hinder other existing security protocols is critical.

DISTRIBUTION:

1	MS 4523	LDRD Office, 04523
1	MS 0451	S.G. Varnado, 06500
1	MS 0784	R.E. Trelue, 06501
1	MS 0785	R.L. Hutchinson, 06516
1	MS 0455	R.S. Tamashiro, 06517
5	MS 0784	M.J. Skroch, 06512
5	MS 0785	J.T. Michalski, 06516
5	MS 0784	R.A. Duggan, 06512
1	MS 0455	D.C. Smathers, 06512
1	MS 9018	Central Technical Files, 8945-1
1	MS 0612	Review and Approval Desk, 12690 For DOE
2	MS 0899	Technical Library, 09616