Identifying Network Packets Across Translational Boundaries

Dr. Napoleon Paxton, Joseph Mathews
Center for High Assurance Computer Systems
US Naval Research Laboratory
Washington, DC 20375

***Abstract*— A translational boundary is any computer network system which performs network address translation in order to act as an intermediary between client requests and server responses. Since boundaries essentially hide networks from the world by acting on their behalf, a sensor monitoring traffic for malicious activity outside of a boundary would attribute the boundary itself as the target of an attack rather than the actual host affected behind the boundary. This challenge is exacerbated inside of tiered network architectures and drives the need for a capability to track network communications across boundaries. While several attempts have been made at addressing this problem space, existing approaches are often difficult to implement or fundamentally problematic. We propose a novel method for tracking communications across boundaries based on the fact that the message being transmitted must remain constant and intact in order for it to be successfully interpreted by a server. The proposed method leverages cryptographic hashing techniques applied towards the application layer payload of network packets from two different perspectives on the network, enabling correlation before and after the packet headers are modified by the boundary. The technique can be implemented atop open source technology on commodity hardware, and provides a stable foundation for building tiered enterprise network architectures with an inherent capability for pinpointing malicious activity.**

Keywords-source identity; translational boundary; packet marking

I. Introduction

A translational boundary, referred to from here on as simply a boundary, is any computer network system that performs network address translation in order to act as an intermediary between client/server transactions. When a client attempts to connect to a server in order to access some resource, the boundary intercepts, alters, and relays the request on behalf of the client. Doing this allows the boundary to impersonate the client and provide transparent routing to the packet's intended destination. The boundary also provides the response back to the client on behalf of the server. The boundary's existence, with few exceptions, is transparent to both the client and server application.

A. Network Address Translation

Boundaries implement a technique known as Network Address Translation (NAT). NAT is an Internet Engineering Task Force standard used to allow multiple computers on a network to share an Internet Protocol (IP) address [1, 2]. It was originally introduced as a means to continue the Internet's growth despite rapid depletion of the IPv4 address space [3]. Its ancillary intent was to hide a network's internal topology and architecture from the world by 1) using unique, discrete address spaces for both the internal and external network segments as well as 2) mediating all inbound and outbound communications between those segments.

B. Communication Through A Translational Boundary

A boundary sits in the path of communication between a client and server so that it can intercept and relay the client's request to the server as well as the server's response to the client. Fig. 1 illustrates the flow of data between a client and server through a boundary.

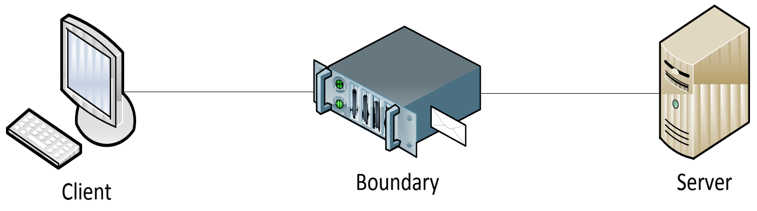


Figure 1 – Topology of a Client/Server transaction through a Translational Boundary

When an application on a client needs to communicate with a service on a server, it will open a network socket in order to transmit its message over an Internet Protocol (IP) network. The network socket is characterized by a source IP address, source port, destination IP address, destination port, and network protocol, which are used to address an IP packet header. The source IP address defines the client while the destination IP address defines the server. Likewise, the source port defines the client application where requests originate, while the destination port defines the service where the server application processes the request and issues a response. The network protocol indicates whether the resulting packet (and

COLLABORATECOM 2014, October 22-25, Miami, United States
Copyright © 2014 ICST
DOI 10.4108/icst.collaboratecom.2014.257685

session) is connection-oriented via the Transmission Control Protocol (TCP) or connection-less via User Datagram Protocol (UDP) [7].

The IP packet, which encapsulates the message, is intercepted by the boundary. The boundary alters the contents of the packet header by replacing the source IP address and source port with its own boundary IP address and boundary-assigned application port. The assignment of a unique application port is used to disambiguate clients' sessions. The boundary actively maintains a translation table of mappings for each connection from every client it serves [4].

Once received and processed, the server addresses its response back to the boundary IP address and boundary-assigned application port, which is the perceived origin of the original request.

The boundary compares incoming response packets to its established translation table. If the server's response packet header has a source IP address (server IP address), source port (server application), and network protocol which match the destination IP address (server IP address), destination port (server application), and network protocol of a record in the translation table, the boundary performs a reverse translation. The boundary replaces the packet's destination IP address (boundary IP address) and destination port (boundary-assigned application port) with the matching record's source IP address (client IP address) and source port (client application).

The boundary transmits the modified packet back to the client. Subsequent packets from the same session are translated in the same manner. It's important to note a supplementary effect of this process is that the boundary prevents unsolicited packets from entering a client's network. Since an unsolicited packet would have no record in the boundary's translation table, it would simply be discarded.

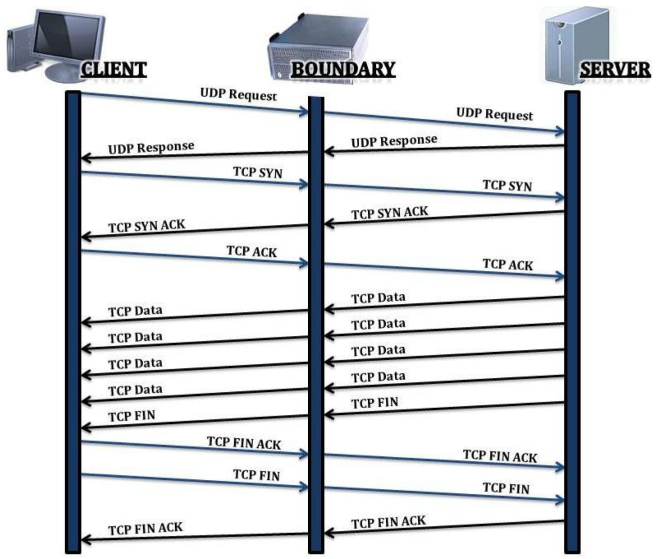


Figure 2 - Connection sequence between a Client and Server through a Translational Boundary

Table 1 - Contents of an IP Packet Header through a Translational Boundary

IP Packet Header

	Source IP Address	**Source Port**	**Dest IP Address**	**Dest Port**
Client Request	Client IP Address	Client app port	Server IP address	Server app port
Bound Request	External Bound IP address	Bound assigned app port	Server IP address	Server app port
Server Response	Server IP address	Server app port	External Bound IP address	Bound assigned app port
Bound Response	Server IP address	Server app port	Client IP address	Client app port

Fig. 2 illustrates a time line of client to server connections for both the UDP and TCP protocols through a boundary. The boundary intercepts, alters and relays packets from the client to the server and vice versa. The sequence of transactions illustrated in **Error! Reference source not found.**, as seen from the packet header's perspective, is enumerated in Table 1. While client requests are sourced from the client, boundary requests are altered to appear from the boundary. Likewise, server responses are addressed to the boundary, whereas boundary responses are altered to appear addressed directly to the client.

C. Types of Network Address Translation

Though there are many implementations found in a myriad of network devices, there are fundamentally two distinct ways of characterizing NAT. They are based upon where the boundary intercepts the transaction between the client and server, or in other words whether the boundary exists on the client's network or on the server's network [6].

When NAT is performed on the client's network, the boundary alters egress communication originating from internal clients. This process is also referred to as "IP Masquerading", "Port Address Translation", "Network Address and Port Translation", "Many-to-One NAT", or "NAT Overload". The boundary hides the entire IP address space of the client's network behind a single unique IP address[1]. The boundary alters the source IP address, the source application ports and

[1] In rare circumstances, clients may map to more than one unique IP address.

their associated checksums within each packet header. A translation table is established and dynamically maintained in order to disambiguate sessions. Fig. 3 illustrates how the source address changes across a boundary. The upper screen capture depicts a session before it was modified by a boundary, whereas the lower screen capture depicts the same session after it was modified by a boundary. While the destination IP address of the server remains constant, the source address was altered from a 172 prefix to a 132 prefix by the boundary. When NAT is performed on the server's network, the boundary alters ingress communication originating from external clients. This process is also referred to as "Port Forwarding", "Static NAT" or "Destination Network Address Translation". The boundary hides the entire IP address space of the server's network behind a single unique IP address. The boundary alters the destination IP address and the associated checksums within each packet header. This translation is performed either through static configuration of the boundary or dynamically through NAT Traversal.

D. Examples of Boundary Devices

Boundaries were traditionally used to connect an isolated network with private unregistered addresses to an Internet with publicly registered and globally unique addresses. The practice of assigning reusable IP addresses in combination with NAT served as an effective tool for alleviating the consequences of IPv4 address exhaustion [1, 2, 3, 6, 7, and 8]. NAT's inherent ability to filter out unsolicited traffic led to its perception as a rudimentary network security mechanism. Boundaries have since become a common, indispensable feature ubiquitously found in a plethora of network devices designed to govern network connections.

- Routers forward packets between computer networks. Address information from a packet header is read by the router and used to direct the packet to its intended destination. This is accomplished through a routing table, which defines what networks are connected to which of its physical interfaces. Though by strict definition routers only pass packets between interfaces, most modern routers have an integrated capability to perform NAT as well as rudimentary packet filtering [9].

- Firewalls regulate packets entering and exiting a computer network based upon some set of security criteria defined by a policy [10]. Modern firewalls can route packets in a similar manner to routers, albeit with fewer interfaces for segments of the network where packets must be inspected. Almost all modern firewalls have the ability to implement NAT [11].

- Network proxies are systems which act as an intermediary between client requests and server responses. Forward proxies act on behalf of a client requesting resources, whereas reverse proxies act on behalf of a server providing resources. Proxies serve a variety of purposes, including but not limited to: content filtering, caching, content modification, insertion, and content customization.

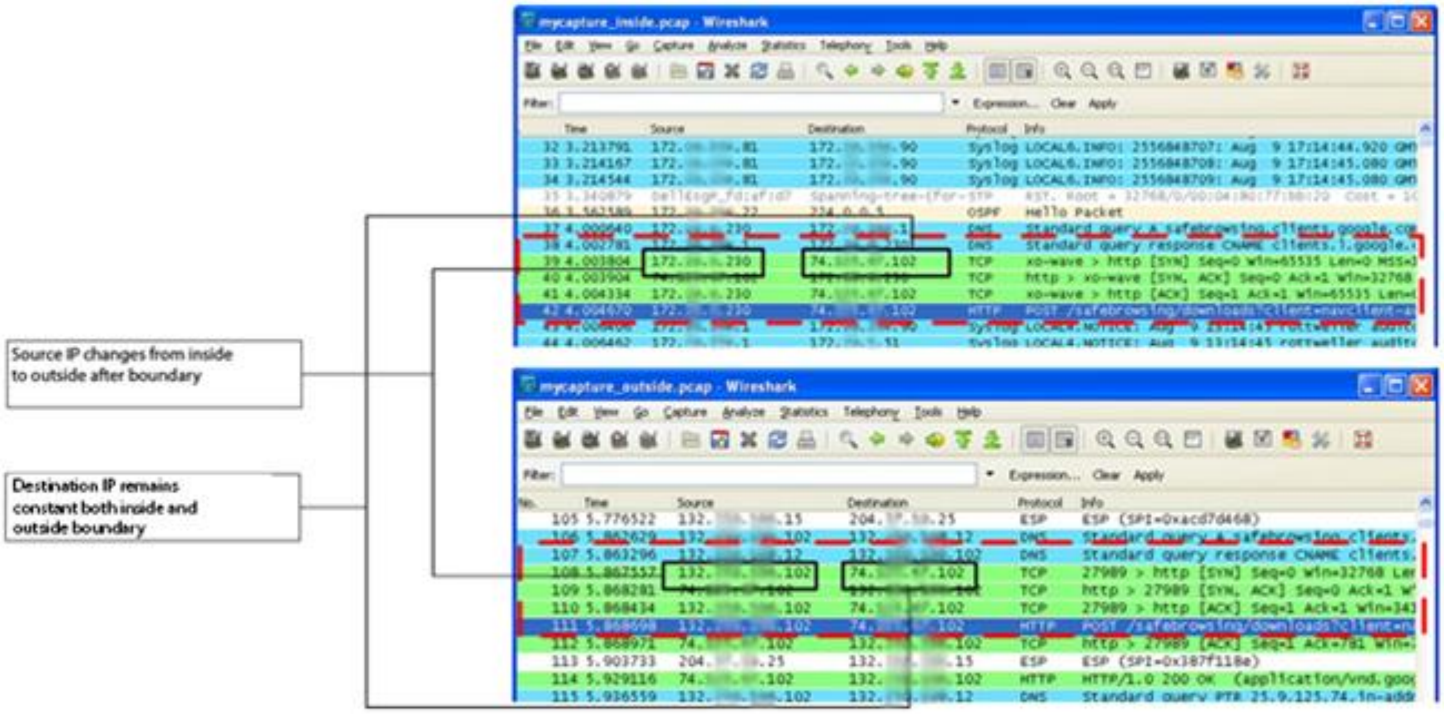
  Proxies further be characterized as transparent, those which are invisible to the client and server application, or non-transparent, those which require the client to know about their existence and embed additional information in the packet payload. For example, a web browser allows a user to manually configure a non-transparent proxy server for which to send its requests to.

- Internet Gateway Devices connect computer networks, usually residential broadband networks, to an Internet Service Provider (ISP) backbone. They can be small hardware appliances such as cable modems or software implementations such as Internet Connection Sharing found in Microsoft Windows XP. They typically implement a combination of features found in firewalls, routers and sometimes proxies.

- IPv6 Gateways connect networks running the v4 implementation of the IP stack (IPv4) to a network running IPv6. The gateway merely translates packets between the stacks, with little regard to which version is run on the internal or external segments of the network. This is opposed to gateways which simply tunnel IPv4 traffic within IPv6 connections. These gateways implement a variant of NAT known as "NAT64" which replaces an IPv6 address on side with an IPv4 address on the other.

- Unified Threat Management Appliances are hardware or software appliances which combine features from all of the above device categories and more. They will usually have enhanced port density in order to accommodate connections between multiple networks, both as a firewall and as an intrusion prevention sensor.

Practically speaking, interception is accomplished through strategic placement in the network architecture (routers, firewalls, transparent proxies, gateways) or static configuration of the client itself (non-transparent proxies).


Source IP changes from inside to outside after boundary
Destination IP remains constant both inside and outside boundary
mycapture_inside.pcap - Wireshark
mycapture_outside.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter: Expression... Clear Apply

```
   Time        Source                Destination           Protocol Info
32 3.213791    172.[redacted].81     172.[redacted].90     Syslog   LOCAL6.INFO: 2556848707: Aug  9 17:14:44.920 GMT
33 3.214167    172.[redacted].81     172.[redacted].90     Syslog   LOCAL6.INFO: 2556848708: Aug  9 17:14:45.080 GMT
34 3.214544    172.[redacted].81     172.[redacted].90     Syslog   LOCAL6.INFO: 2556848709: Aug  9 17:14:45.080 GMT
35 3.340879    Dell(sgP_fd:af:d7     Spanning-tree-(for-   STP      RST. Root = 32768/0/00:04:80:77:b8:20  Cost = 1
36 3.562589    172.[redacted].22     224.0.0.5             OSPF     Hello Packet
37 4.000640    172.[redacted].230    172.[redacted].1      DNS      Standard query A safebrowsing.clients.google.co
38 4.002781    172.[redacted].1      172.[redacted].230    DNS      Standard query response CNAME clients.l.google.
39 4.003804    172.[redacted].230    74.[redacted].102     TCP      xo-wave > http [SYN] Seq=0 Win=65535 Len=0 MSS=
40 4.003904    74.[redacted].102     172.[redacted].230    TCP      http > xo-wave [SYN, ACK] Seq=0 Ack=1 Win=32768
41 4.004334    172.[redacted].230    74.[redacted].102     TCP      xo-wave > http [ACK] Seq=1 Ack=1 Win=65535 Len=
42 4.004670    172.[redacted].230    74.[redacted].102     HTTP     POST /safebrowsing/downloads?client=navclient-a
43 4.00[illegible] 172.[redacted].1  172.[redacted].90     Syslog   LOCAL4.NOTICE: Aug  9 13:14:45 Fortweiler auditd
44 4.006462    172.[redacted].1      172.[redacted].31     Syslog   LOCAL4.NOTICE: Aug  9 13:14:45 Fortweiler auditd
```

```
No.  Time      Source                Destination           Protocol Info
105 5.776522   132.[redacted].15     204.[redacted].25     ESP      ESP (SPI=0xacd7d468)
106 5.862629   132.[redacted].102    132.[redacted].12     DNS      Standard query A safebrowsing.clients.
107 5.863296   132.[redacted].12     132.[redacted].102    DNS      Standard query response CNAME clients.
108 5.867557   132.[redacted].102    74.[redacted].102     TCP      27989 > http [SYN] Seq=0 Win=32768 Len
109 5.868281   74.[redacted].102     132.[redacted].102    TCP      http > 27989 [SYN, ACK] Seq=0 Ack=1 W
110 5.868434   132.[redacted].102    74.[redacted].102     TCP      27989 > http [ACK] Seq=1 Ack=1 Win=343
111 5.868698   132.[redacted].102    74.[redacted].102     HTTP     POST /safebrowsing/downloads?client=n
112 5.868971   74.[redacted].102     132.[redacted].102    TCP      http > 27989 [ACK] Seq=1 Ack=781 Win=
113 5.903733   204.[redacted].25     132.[redacted].15     ESP      ESP (SPI=0x387f118e)
114 5.929116   74.[redacted].102     132.[redacted].102    HTTP     HTTP/1.0 200 OK  (application/vnd.goo
115 5.936559   132.[redacted].102    132.[redacted].12     DNS      Standard query PTR 25.9.125.74.in-addr
```

Figure 3 – Source IP is altered by the boundary and Destination IP remains constant

Boundaries serve a variety of purposes: They provide a cost effective method of connecting multiple hosts to a network under one Internet Protocol (IP) address. They can boost the perceived speed of the network by caching content for their clients. They can enforce security policy on network services or application layer content. They can audit and log network activity. They can try to detect malicious content. These attributes account for the pervasiveness of boundaries, both as a functional component of network architectures as well as a network security mechanism.

E. Tiered Network Architectures

Many enterprises employ tiered architectures, where the network can be topologically described as comprised of hierarchically interconnected enclaves [4]. This concept is illustrated in Fig. 4. The core layer is comprised of assets which form the backbone infrastructure of the network. The distribution layer interconnects the access layer to the core layer. The access layer provides connectivity to clients. This model may be applied towards networks of all sizes. The model is purely notional and need not be limited to three layers.

In a simple, physically collocated network such as a single building, the core layer would be comprised of a backbone router which serves as the Internet gateway. The access layer could be comprised of switches for each connected story of the building. The distribution layer in turn would be comprised of switches (or VLANs) to connect individual offices to the access layer.

In a complex, geographically diverse network, the core layer could be comprised of a large routing infrastructure. The access layer would serve as a step site to provide connectivity to geographically disparate points of presence. The distribution layer in turn would provide access to clients on a particular location [12].

Since each layer may represent independent enclaves of the Enterprise, each enclave may employ its own boundary device for any number of reasons discussed in the previous section. For example, consider a global network where the core, distribution and access layers are interconnected through leased circuits from an ISP. Since each layer has to pass traffic through the Internet, it is often in their best interest to employ a boundary protection mechanism.

II. CHALLENGES

NAT fundamentally breaks the originally envisioned model of end-to-end connectivity across the Internet. Boundaries hide their clients from the world by acting on their behalf. When a client communicates with a server, the server perceives the boundary as the client, rather than the actual client. This is true regardless of whether the communication itself could be characterized as benign or malicious.

The detection of malicious activity on a network is primarily accomplished through a myriad of sensing technologies [5]. Network Intrusion Detection/Prevention sensors inspect network traffic in order to identify and alert on policy violations or malicious activity. Whereas firewalls deny

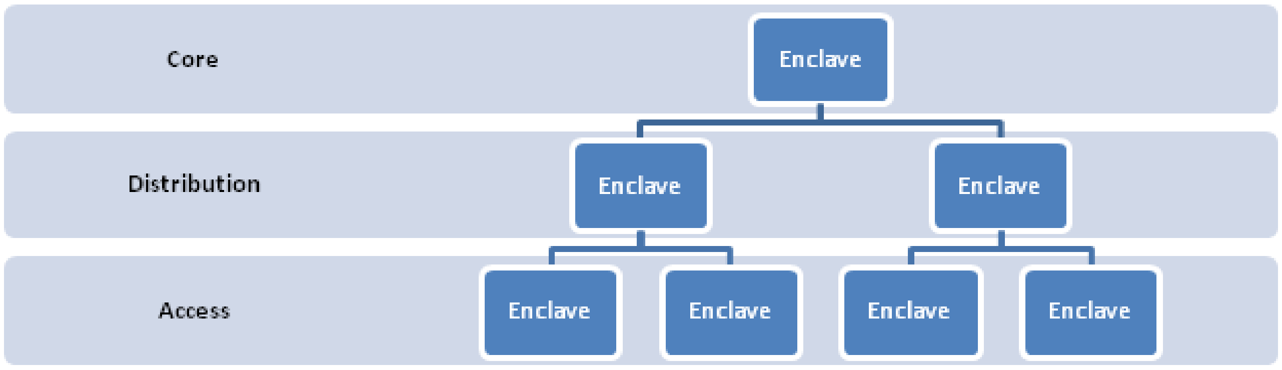


Figure 4 - Tiered Network Architecture

traffic by default and allow by exception, sensors allow traffic by default and deny by exception. Positive identification of malicious activity is accomplished through signatures that identify unique patterns found in network traffic indicative of vulnerabilities or exploits.

Active sensing technology (intrusion prevention) accesses network traffic through inline placement on the network. Passive sensing technology (intrusion detection) accesses network traffic through passive mirroring of traffic via a hub, tap, or port mirroring switch. In either case, sensor placement is a strategic concern in the design of secure network architectures [5, 11].

If packets were intercepted between the client and boundary, one would expect to observe the true address of the client and server. If the packets were intercepted between the boundary and the server, one would expect to observe the address of the boundary and server. Applied to tiered network architectures, inspecting packets at the core or distribution layers will not provide adequate visibility into the particular client communicating at the access layer, since they are in effect hidden by the boundary at that layer. For example, consider a home network comprised of a desktop PC, laptop, tablet, phones, and other network/Wi-Fi enabled devices. All devices sit behind an Internet Gateway Device provided by the ISP which performs NAT. The ISP for the residential network would not be able to easily pinpoint which device is communicating since all communications appear to be sourced from the gateway device.

This problem is further exacerbated in distributed networks where each subordinate tier runs its own boundary. The distribution layer would have inadequate visibility into the access layer, and the core layer would have inadequate visibility into both the distribution and access layers. While this challenge can be solved through adequate placement of sensors, there are usually external factors such as bandwidth limitations and enclave ownership which prevent full sensor coverage.

This is a significant problem for the defence of networks with tiered architectures, especially since an introspective look behind a translational boundary violates the security policy in almost all cases. When malicious activity is detected, the first step in an incident response protocol is to gauge the scope and impact of the incident by identifying the relevant parties involved. One cannot isolate the affected hosts if they are hidden by a translational boundary.

III. Current Approaches

The challenge of identifying the true source of an event on the network falls in to the realm of a larger problem space known as the IP trace-back problem. Most solutions for this problem are expensive to implement or fail to accurately attribute activity behind boundaries.

First and foremost, any discussion around NAT almost inevitably yields an immediate response indicating how fully ubiquitous adoption of IPv6 will supersede any continued need for NAT. While it is true that homogenous adoption of IPv6 and the subsequent deprecation of IPv4 will theoretically render NAT obsolete, it does not address the reality of networks that run dual stack. Any general purpose IP protocol stack that supports IPv6 will also support IPv4 in order to ensure any two parties can have a conversation, even if one of them only speaks IPv4. Since running dual stack is a key milestone to full adoption of IPv6, NAT is absolutely critical in order to provide every network interface both an IPv6 and an IPv4 address [13]. This is so because IPv6 possesses a significantly larger address space than IPv4, whereas addresses for both stacks are needed. IPv6 addresses this through its specification for carrier grade NAT (sometimes referred to as large scale NAT).

Probabilistic packet matching involves setting aside a fixed number of unused or rarely used bits in the packet header (or an entirely new packet) in order to mark packets as they traverse routers [14, 15]. The first problem with this approach is that the vast majority of routers that make up the Internet are commercial-of-the-shelf devices, which do not implement custom capabilities such as packet tagging. Moreover, while there are a number of statistical approaches on tagging methods, each approach requires a fixed number of packets to be observed before accurately reconstructing a path. More often than not, the number of packets is usually high, limiting its practical effectiveness to detecting distributed denial of service or flooding attacks. Further, approaches which require tagging packets with the IP address of the router traversed by the packet do not provide much value added in tiered networks that heavily leverage private IP addresses. Packets may traverse multiple routers with the same reusable private IP addresses. The final drawback of this approach is mark spoofing, where an attacker intentionally injects packets with known false tags in order to throw off path reconstruction algorithms.

Deterministic packet marking operates similarly to probabilistic packet matching, with the exception that packets are tagged with the network's ingress address. That is to say, it treats router interfaces as an index for trace-back as opposed to the router itself. The largest advantage of this approach is that it separates inbound and outbound packets, accounting for asymmetrical route paths [16]. While deterministic packet marking generally requires fewer packets to be observed in order to reconstruct a path, it still requires a custom tagging capability not readily available in commercial off the shelf devices [17].

State table analysis involves exposing the translation table generated by a boundary to the outside world. If the state table, which contains an index of the true client for each and every transaction, were exposed it would enable ad-hoc queries in order to identify the true source IP address or source application port of an open session [18]. There are several problems with this approach. First off, most boundaries are commercial devices and purposely do not support such a capability in light of performance considerations. Next, the contents of the state table are highly dynamic and change frequently based upon how many active sessions they are translating. Closed sessions are expired from the state table, rendering any human driven ad-hoc queries futile. Finally, exposing a service interface on a boundary, which typically serve as protection mechanism for networks, open up a potential attack vector for external entities unless properly mitigated through separation of the management and data paths.

Time-To-Live (TTL) analysis focuses on inferring path by observation of the IP protocol TTL field. TTL indicates how many hops a packet may route through before being expired and discarded. Each IP forwarding device reduces the value of the TTL field by 1 before relaying the packet to its next hop. The value of this assignment depends upon the implementation of the network stack [19]. In Microsoft Windows operating systems, it is usually 128, whereas on Linux operating systems, it is usually 64. If a client is directly connected to an ISP, then the ISP's edge router will always see TTL values of 128 (or 64). If there is a boundary in the path, the ISP will see TTL values of 127 (or 63). This approach provides little value for packets which are routed through multiple hops and does nothing for identifying their true source address. Moreover, there are implementations of NAT which do not modify the TTL field on purpose in order to remain stealth.

Similarly, IP identifier analysis makes it possible to ascertain whether or not a client is a boundary or not, as well as estimate how many clients they are serving. When a client originates an IP packet, it creates a 16 bit ID number in the ID field of the IP packet header. ID numbers from a single client are usually assigned in sequence. If multiple clients are multiplexed through a boundary, a unique number of sequences will be observed, one for each client transaction. Inferring the number of sequences may serve to estimate how many clients the boundary services. Again, this will not aid identification of a client's true source address.

IV. Proposed Approach

Despite the challenge of a boundary replacing a client's network address with its own, there is a constant for the majority of packets: the actual message being transmitted, or the application layer payload of the packet. The application layer payload must remain completely intact both inside and outside of the boundary in order for a packet to be successfully received and interpreted by a server. Since the payload remains constant between the client and server, we hypothesize and validate that it can also be used as a unique identifier in and of itself.

Our proposed solution leverages cryptographic hashing techniques applied towards the application layer payloads of network packets from at least two different perspectives on the network (both sides of the boundary). By analysing the payload from two different perspectives, one can match an observed packet before and after it is modified by a translational boundary. Fig. 5 presents a framework to accomplish this.

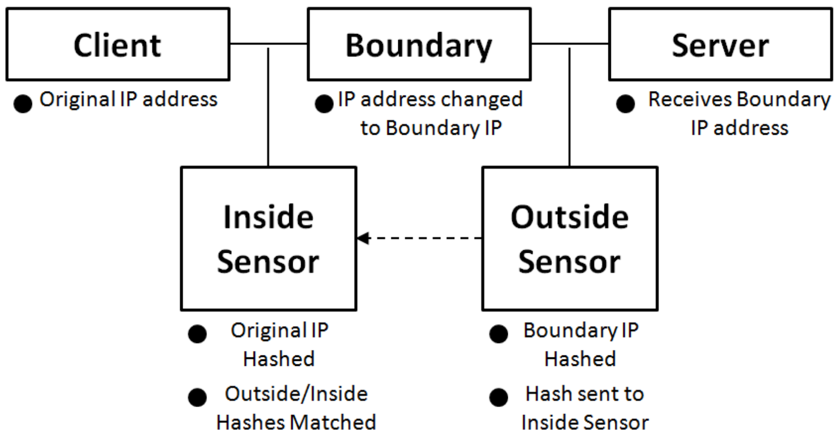

Client
Original IP address
Boundary
IP address changed to Boundary IP
Server
Receives Boundary IP address
Inside Sensor
Original IP Hashed
Outside/Inside Hashes Matched
Outside Sensor
Boundary IP Hashed
Hash sent to Inside Sensor

Figure 5 - Framework Diagram

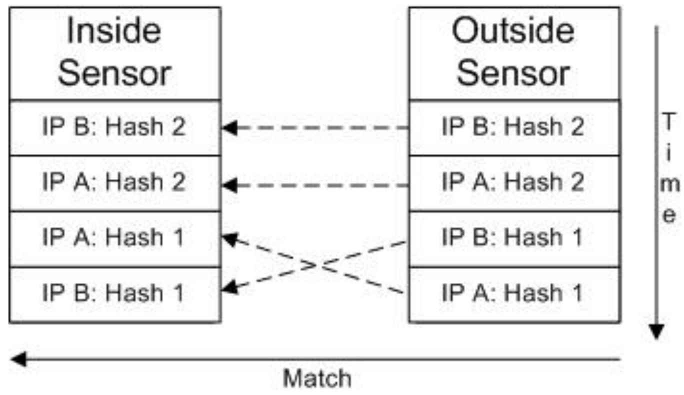


Figure 6 - FIFO approach to hash matching

A. Cryptographic Hash Calculation

We rely on two commodity servers running full packet capture in promiscuous mode via open source software. While a single server with two network interfaces would suffice, the process was implemented in a distributed fashion in order to scale to the demanding requirements of full packet capture, especially on high bandwidth links. The first server is referred to as "inside", which passively records traffic on the client network, before the contents are altered by a boundary. The second server is referred to as "outside", which passively records traffic externally, after it has been modified by the boundary. Cryptographic hashes are independently calculated on both sides of the boundary. Results are only correlated after all the packets from both sides have been analyzed.

In theory, any cryptographic hashing algorithm can be used to identify unique payloads. The hash selected for proof of concept implementation is MD5 hash. MD5 was chosen because it is relatively easy to compute, it does not require a large memory footprint, and it produces a unique resulting value.

The hash value from each payload is stored in a database along with the IP address and timestamp of when it was sensed. While this occurs on both the inside and outside sensors, a separate process mirrors the contents of each server's database into a single instance on the inside sensor. This is done in order to construct a unified location for data in order to match payloads.

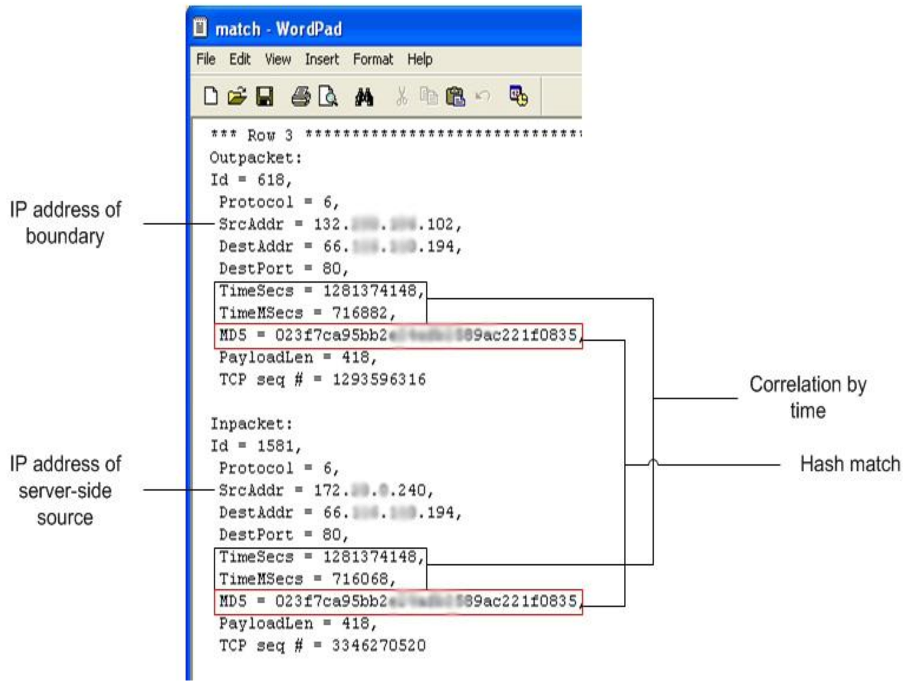
B. Correlating Results

Payloads are matched based on three criteria; hash, time, and IP address. When an identical hash is observed on the outside and inside, there is a high probability that the hashes belong to the same payload. A First-In-First-Out approach is leveraged in order to match outside and inside hashes with respect to their observed timestamp. After a hash is observed on the outside, the closest matching hash (with respect to the timestamp) on the inside is identified as the corresponding match.

Fig. 6 depicts an example of four packet payloads that have been hashed by both the inside and outside sensors. Here we show that after the outside hash is computed, the match program will search for the first hash from the inside it can find that occurs after the outside value was computed.

C. Proof of Concept

A proof of concept implementation was constructed in the C programming language. Data is stored and retrieved from SQL databases, and the entire framework is operated atop commodity hardware. The proof of concept has two operating modes: live packet capture and file upload. Live packet



```
*** Row 3 ******************************
Outpacket:
Id = 618,
 Protocol = 6,
 SrcAddr = 132.[redacted].[redacted].102,
 DestAddr = 66.[redacted].[redacted].194,
 DestPort = 80,
 TimeSecs = 1281374148,
 TimeMSecs = 716882,
 MD5 = 023f7ca95bb2[redacted]89ac221f0835,
 PayloadLen = 418,
 TCP seq # = 1293596316

Inpacket:
Id = 1581,
 Protocol = 6,
 SrcAddr = 172.[redacted].[redacted].240,
 DestAddr = 66.[redacted].[redacted].194,
 DestPort = 80,
 TimeSecs = 1281374148,
 TimeMSecs = 716068,
 MD5 = 023f7ca95bb2[redacted]89ac221f0835,
 PayloadLen = 418,
 TCP seq # = 3346270520
```

Figure 7 - One match generated by our framework

capture allows the user to generate hash values from packet payloads observed on the network interface in near real-time. File upload hashes packet payloads from a capture file that has been previously recorded and stored on a file system. Figure 7 shows one match from the output of the framework.

D. Challenges

The proposed approach for matching cryptographically hashing payloads has one critical assumption: that the payloads sensed both inside and outside are identical. If either payload has been altered in any way, the computed hash will not be the same and therefore will not match. One example of this is non-transparent proxies which make slight modifications to the payload in order to do things such as media type transformation, protocol reduction, or anonymity filtering [20].

A potential alternative approach would be to leverage different classes of hashing techniques in order to account for slight variations in payload alterations. For example, fuzzy hashing may be able to match payloads that have been slightly altered, as in the case of non-transparent proxies or deep packet inspection platforms. Fuzzy hashing is similar to traditional cryptographic hashing; with the exception that it produces a result value that is reflective of how similar the original data is to the altered data.

V. Conclusion

The ability to identify the true source of packet transmission through a boundary will provide significant benefits to network security. Although this is not an end-all solution to the origin discovery of malicious content, it does provide a way to quickly identify nodes that are infected with malicious content which will allow the network administrator to better identify the scope of the malicious incident.

The process we display here is highly modular and can be implemented atop open source technology on commodity hardware, and provides a stable foundation for building tiered enterprise network architectures with an inherent capability for attribution of malicious activity. Enterprises with significant visibility and monitoring investments into the network backbone can utilize this technique to attribute malicious activity sensed at the edge of a network back to its original source.

References

[1] Network Working Group (1994). RFC 1631: The IP network address translator (NAT). http://tools.ietf.org/html/rfc1631

[2] Network Working Group (1999). RFC 2663: IP network address translator (NAT) terminology and considerations. http://tools.ietf.org/html/rfc2663

[3] Network Working Group (1996). RFC 1918. address allocation for private Internets. http://tools.ietf.org/html/rfc1918

[4] Donohue, Gary A. (2011). Network warrior, 2nd Edition. O'Reilly Media, Inc.

[5] Bejtlich, R. (2004). The tao of network security monitoring. Addison Wesley.

[6] Davidoff, Sherri & Ham, Jonathan. (2012). Network forensics: tracking hackers through cyberspace. Prentice Hall.

[7] Stevens, R.W. (1994). TCP/IP illustrated, volume 1: The Protocols. Addison Wesley.

[8] Microsoft Technet. (2005). Overview of network address translation (NAT) in Windows XP. | http://technet.microsoft.com/en-us/library/bb457077.aspx

[9] RFC 1812. (1995). Requirements for IP version 4 routers. https://tools.ietf.org/html/rfc1812

[10] Cheswick, W.R, Bellovin, S.M., Rubin (2003). Firewalls and Internet security: repelling the wily hacker. Addison-Wesley Longman Publishing Co, Inc Boston MA.

[11] Ackerman, R., Roedig, U. and Steinmetz. (2004). Evaluating and improving firewalls for IP-telephony environments. IP Telephony Workshop (IPTel).

[12] Fowler, M. (2002). Patterns of enterprise application architecture.

[13] Mullins, R. (2012). Deploying dual-stack IPv4 and IPv6networks.

[14] Goodrich, M.T. (2002). Efficient packet marking for large-scale IP traceback. ACM CCS. Washington DC.

[15] Li, J., Sung, M., Xu, J., Li, L. (2004). Large-scale IP traceback in high-speed Internet: practical techniques and theoretical foundation. Proceedings IEEE Symposium on Security and Privacy.

[16] Belenky, A., Ansari, N. (2003). IP traceback with deterministic packet marking. IEEE Communications Letters Vol 7, no 4.

[17] Park, K., Lee, H. (2000). On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack. Computer Science Technical Reports, Paper 1491, http://docs.lib.purdue.edu/cstech/1491

[18] Afanasyev, A., Tilley, N., Longstaff, B.,Zhang, L. (2009). BGP routing table: trends and challenges .http://lasr.cs.ucla.edu/afanasyev/data/files/Afanasyev/BGP%20Routing%20Table%20Trends%20and%20Challenges.pdf

[19] Zander, S., Branch, P, Armitage, G. (2007). Error probability analysis of IP time to live covert channels. IEEE Communications and Information Technologies.

[20] Rabinovich, M, and Spatscheck, O. (2002). Web caching and replication. Addison Wesley; 1st edition