Long-Term Sessions: this is why we can't have Nice Things

Steve Ocepek
Senior Security Consultant
SpiderLabs

June 24, 2009

A Whitepaper for Black Hat USA 2009

Table of Contents

1 Introduction

The idea of detecting network attacks based on behavior offers numerous benefits, but has also been the subject of much debate. Often, the process of “teaching” these systems is costly and prone to error. The lack of proper configuration, however, leads to false positives that serve only to frustrate those that the system is supposed to protect. These issues frequently lead to the abandonment of behavioral techniques in favor of signature-based detection methods.

On the other hand, the HTTP protocol is quickly replacing other protocols as the de facto method of Internet communication. With this convergence comes a stronger, better-defined behavioral model. Normal behavior is easier to define: truly, many network administrators are limiting Internet access to ports 80 and 443 with few exceptions due to the acceptance of HTTP. And because HTTP is essentially a session-less protocol, many of these valid communication streams start and end quickly and definitively.

While valid sessions tend to be short and transactional, forensics data points to long-term sessions as a source of unauthorized access. Indeed, many exploits involve the creation of a long-term session, many times initiated by the victim, that enables the attacker to execute remote commands at will. Watching a production web server initiate a connection to a remote cable modem address would surely raise the eyebrows of most security professionals. That being said, many long-term connections are legitimate, instant messaging and large file downloads among them.

This paper describes a method of using session duration for the purposes of detecting unauthorized long-term sessions. Included are examples of common long-term sessions and ideas about how to use public information to discern normal sessions from potentially threatening ones. With luck, the ideas presented here will encourage other ideas, which will ultimately lead to the acceptance of another layer of perimeter security.

2 Typical HTTP Behavioral Model

Details of the HTTP protocol itself are well known, and are covered in depth in RFC 2616. For the purposes of this paper we will examine the common behaviors associated with HTTP and HTTPS, the protocols that provide the foundation for web communications.

2.1 Traditional web browsing

Traditional web browsing can be looked at as a sequence of short-lived transactions across numerous web servers. Starting from the point at which a web page is resolved by DNS into an IP address, the following steps are performed:

1. Host sends a TCP packet with the SYN flag set to Server on port 80 (HTTP) or 443 (HTTPS)
2. Server replies with a TCP packet with SYN and ACK flags set, acknowledging the communication request.
3. Host sends a TCP packet with the ACK flag set to Server, completing the "three-way handshake"
4. Host issues an HTTP GET request for the desired content
5. Server sends requested content
6. Host processes content
7. Host issues follow-up requests for each item that is required to display content, including graphics and rich media (for example Flash byte code, Java applets)

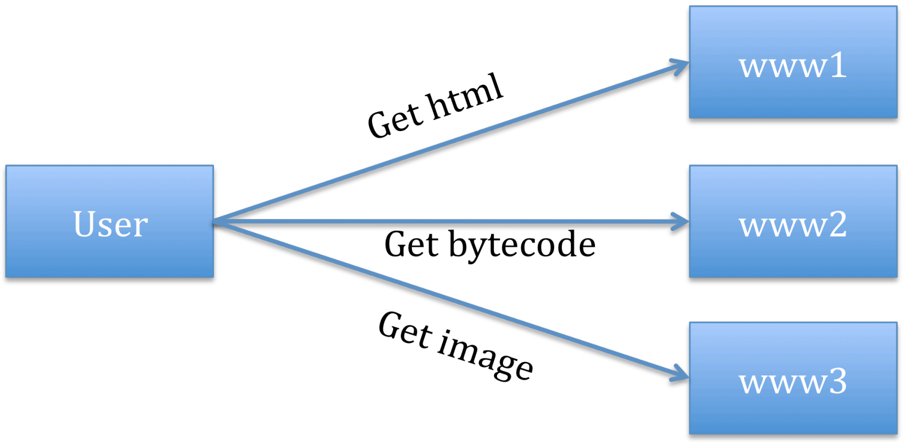

Figure 1. General HTTP Behavior

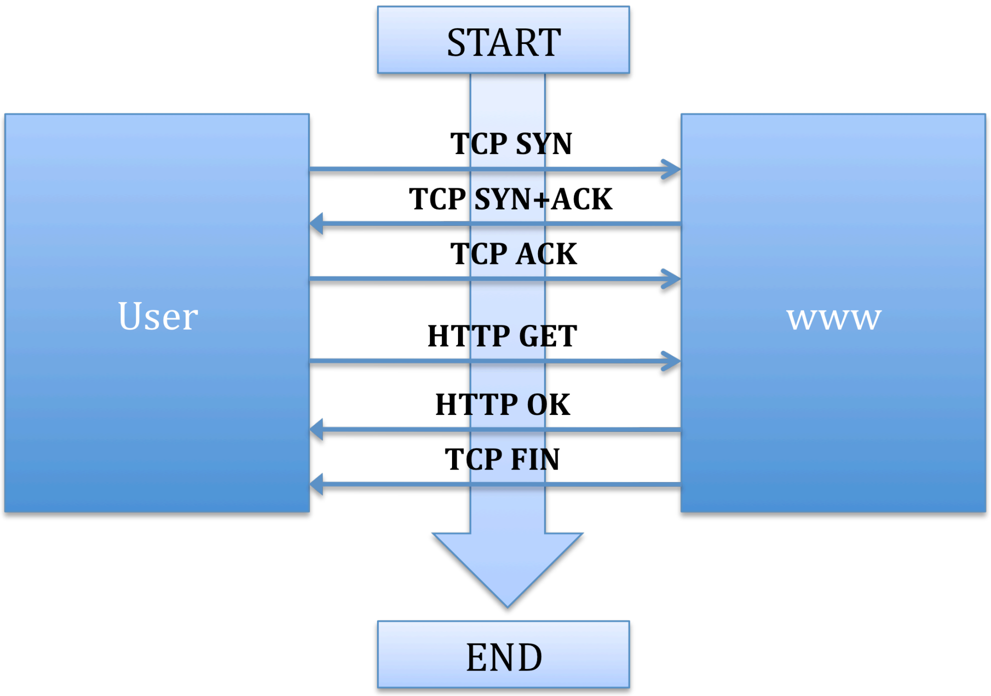

Figure 2. Typical HTTP Short-Lived Session

From a behavioral perspective, it is interesting to note the transactional nature of this sequence. Traditional web browsing is, at its core, a request/response process. Each time another resource is requested, another connection is made, the data is collected, and the communication stream is shut down.

It is important to note that multiple files may be transferred over the same connection. An optimization called "keep-alive" was first implemented under HTTP 1.0 and officially adopted as the default method of session management in HTTP 1.1. By clustering multiple file transfers together into the same HTTP session, a noticeable speed increase is gained. These connections are typically closed within fifteen seconds after the file transfers are complete.

2.2 Long-term web sessions

2.2.1 Comet

As web technologies evolve, a number of technologies are being used to create more responsive, event driven applications. Using a number of technologies commonly referred to as "Comet", some web applications are beginning to maintain long-term connections with clients in order to push data asynchronously.

Examples of this behavior can be found in a number of web applications including web-based chat applications offered by Facebook, Gmail, and Microsoft, among others. Implementations vary - some using a technique called "Long-polling" instead of leaving a session open for the duration. Long-polling will frequently recycle connections during periods of inactivity, causing it to act more like traditional web communications. Authorized applications should be examined to fully understand their behavioral model before creating acceptable use policies.

2.2.2 HTTP File Downloads

Another common source of long-term web sessions is file downloading. Servers that offer large file downloads should be identified and treated differently from the standpoint of behavior modeling. Download servers are commonly separated from those that offer web content, which can make this process easier.

3 Common Long-term Sessions

Though modern applications increasingly rely on web technologies for delivery, a number of common applications still exist that maintain long-term sessions. This section highlights a number of these applications. Each application should be considered by the security staff to determine whether each type of long-term session is authorized.

3.1 Instant Messaging

The asynchronous nature of instant messaging applications makes long-term sessions almost essential. Typically, these applications connect to a central server cluster that connects users with one another. Some features exist to establish connections directly with other users, bypassing the server; this option is usually reserved for direct file transfer or another specific service.

Popular applications in this area include:

- AOL Instant Messenger
- Yahoo! Messenger
- Windows Live Messenger (formerly MSN Messenger)
- Google Talk
- Internet Relay Chat (IRC)

3.2 Peer-to-Peer

Applications that rely on a distributed, unpredictable architecture are referred to as Peer-to-Peer. These applications can establish long-term connections with any number of participants in the distributed network for the purposes of sending and receiving data. Some applications in this space rely on a central point of contact to coordinate peer connections, while others support more discovery-oriented methods.

Examples of commonly used Peer-to-Peer applications include:

- Skype
- BitTorrent
- eDonkey
- Kazaa

3.3 Remote Terminal

This category includes remote text-based applications used by administrative personnel to manage network devices. Popular applications include:

- OpenSSH
- PuTTY
- SecureCRT
- Telnet

3.4 Remote Control

Also used by administrative staff, this category includes applications that allow users to access their desktops from home as well. Some applications in this category are the subject of security concern, and can be detected using the methods described in this paper.

Common applications used by administrators include:

- Microsoft Remote Desktop
- VNC
- X11
- Citrix Access Gateway

Applications that focus on end users include:

- GoToMyPC
- LogMeIn
- PCAnywhere

4 Threats

This section outlines a number of threats that take advantage of long-term sessions – the very things that this paper aims to combat.

4.1 Bind Shells

After exploiting a particular vulnerability on a server, whether through the use of a buffer overflow attack or another form of remote code execution, it is common to load a Bind Shell onto the victim machine. A Bind Shell gives remote access to the attacker, allowing them to execute remote commands at will. This type of asynchronous access requires a long-term session to function.

Roughly two types of Bind Shells exist. A normal Bind Shell opens a port on the exploited device that allows the attacker to connect to it. Reverse Bind Shells operate by connecting back to the attacker's machine, which is necessary in cases where the victim machine is behind a firewall. Understanding these different types of connections illustrates the need for two-way monitoring of sessions. Simply monitoring incoming sessions would not catch Reverse Bind Shells that have been deployed on the local network.

In addition to text-based Bind Shells, some types support remote graphical sessions using protocols such as VNC. A large number of Bind Shells can be found under Payloads in the Metasploit Framework.

4.2 Botnets

Botnets are networks composed of compromised machines that act as "bots". Each of these machines is accessible remotely through a command and control channel. Internet Relay Chat (IRC) is commonly used as a meeting place between bots and "bot herders", who can remotely control each machine.

Long-term sessions are essential to this architecture. Since the infected machines are connecting to an Internet resource, possibly through a firewall, command and control mechanisms must keep the connection alive if at all possible. IRC takes care of this through the use of PING commands, while other implementations may vary.

Depending on the level of access gained by the attacker, any number of programs may be loaded on the infected systems. Keyloggers, network sniffers, and remote control software represent examples of Botnet malware.

4.3 Data Sniffers

During forensic investigation into a number of recent credit card breaches, the investigators found evidence of data sniffers, programmed to recognize credit card numbers and send any matching data to the attackers machine. Whether a long-term session is established or not in each specific implementation, the initiation of this type of connection by a critical server to a remote, unknown resource should be scrutinized regardless. In the next section of the paper, methods for creating policies based on device role are explored.

5 Methods for Detection

This section describes strategies for classifying and detecting long-term sessions.

5.1 Definitions

There are many types of sessions. Though they are all used to transfer data, the source, or initiator, and the port numbers associated with a session can tell us much about the purpose and validity of the session.

The following definitions use the terms “Local” and “Remote” to distinguish host location. For the purposes of this paper, Local devices live inside of a network that we control, even if these hosts span several different geographic locations. Remote devices are unmanaged devices on the Internet.

5.1.1 Local Source

Local hosts generally serve one of two roles. Servers allow connections and provide data to Local Users and Remote devices. User devices, on the other hand, are consumers of the resources provided by Local Servers and Remote devices. A Local Source session is a session that is initiated by one of these two device types.

5.1.1.1 Server type

For the purposes of this paper, a Server is any device that is authorized to accept incoming connection requests. Whether the device exists behind a firewall in a DMZ, or is directly exposed does not matter. When identifying Servers, we are mainly concerned with connection behavior. The type of service provided should also be examined for typical session duration. Web servers generally provide sessions of short duration, whereas FTP server sessions are typically much longer.

Servers rarely function as legitimate Local Sources. When they do, authorized behavior is often specific and well defined.

5.1.1.2 User type

A User device is one that does not accept incoming connections from Remote devices. A consumer of resources, this type of device exists primarily to serve a single user or group. User devices frequently create sessions with Remote Hosts, making them the primary type of Local Source.

5.1.2 Remote Source

A Remote device is an unmanaged Internet entity. Remote Sources are devices that have initiated a session with a Local device. Remote Sources sessions are frequently established with Local Servers, though incoming connections to Local User devices should be rare or non-existent.

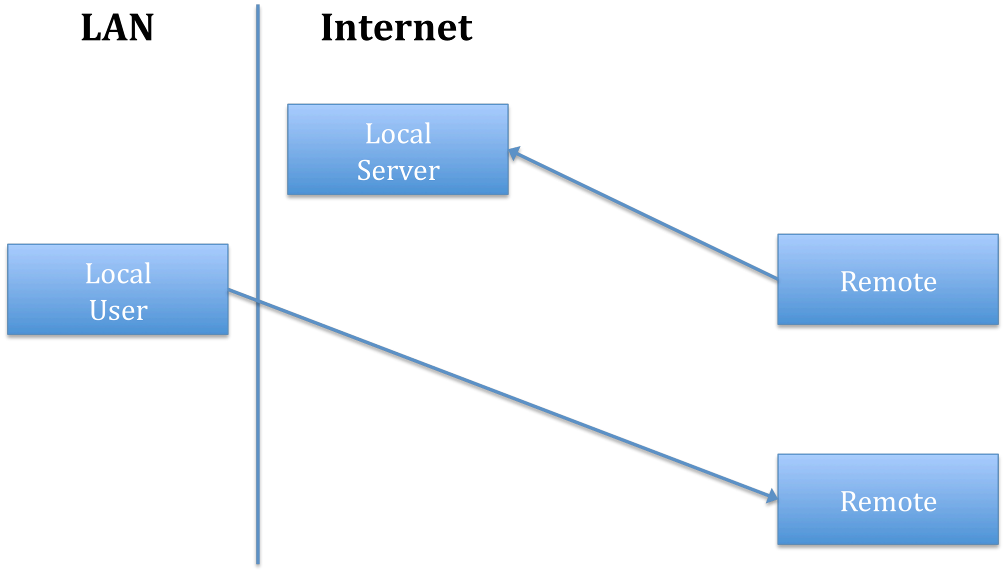

Figure 3. Relationships between Local User, Local Server, Remote

5.2 Tracking Sessions

Current network sessions can be monitored on any device with a modern operating system, but this information is limited in scope to a single device. To truly get a picture of the sessions being maintained by the network as a whole, we need to use a well-placed network sniffing application. The logic used in this application is described here.

Once launched the application begins to sniff network traffic. As the network sniffer collects data, the application begins to identify sessions. Once identified, data about each session is stored in a session record and compared against a defined policy. Each session record contains the following data:

- Timestamp of session start (or first observation, in the case of existing sessions)
- Type
    - Local Source
    - Remote Source
    - Unknown
- Local address
- Remote address

5.2.1 New Sessions

New sessions are identified by the three-way TCP handshake of SYN/SYN+ACK/ACK. The following illustration shows decoded packets that match this pattern.

```
15:00:35.132175    IP    192.168.1.126.52950    >    192.168.1.1.22:    S
2003293573:2003293573(0)      win      65535      <mss      1316,nop,wscale
3,nop,nop,timestamp 809614749 0,sackOK,eol>
```

```
15:00:35.133810    IP    192.168.1.1.22    >    192.168.1.126.52950:    S
2746725458:2746725458(0)    ack    2003293574    win    5792    <mss
1460,sackOK,timestamp 1790130394 809614749,nop,wscale 6>

15:00:35.133853  IP  192.168.1.126.52950  >  192.168.1.1.22:  .  ack  1  win
65535 <nop,nop,timestamp 809614749 1790130394>
```

Figure 4. Three-way Handshake Packet Analysis

When this pattern is detected, the application creates a session record and begins tracking the connection.

5.2.2 Existing Sessions

Detection of existing sessions is more difficult than new sessions, but vital to the application's task. Malware might already have established a connection before the application is started, and should be detected if at all possible. Open connections are detectable by the presence of acknowledgement (ACK) packets between two devices, but the originator of the session must be inferred. The port number used by each device provides a good indication of where the connection started.

5.2.2.1 Port Guessing

The first method of source detection involves a guess based on the port number itself. Using well-known services as a guide, the application can be configured to determine the source based on the ports used by each side of the connection. For example, if Device A is using port 80 and Device B is using port 30453, Device B is very likely to be the source. This is because port 80 is well known for providing web traffic, whereas port 30453 is obscure. Device B therefore likely connected to Device A.

To implement this type of source detection, the application reads a configuration file that lists well-known ports in preference order. The first number in the list that matches one of the ports used in the session is used to denote the server, determining the unmatched port in the session as the source. This prevents conflict in the case that two well-known port numbers are observed inside of a session.

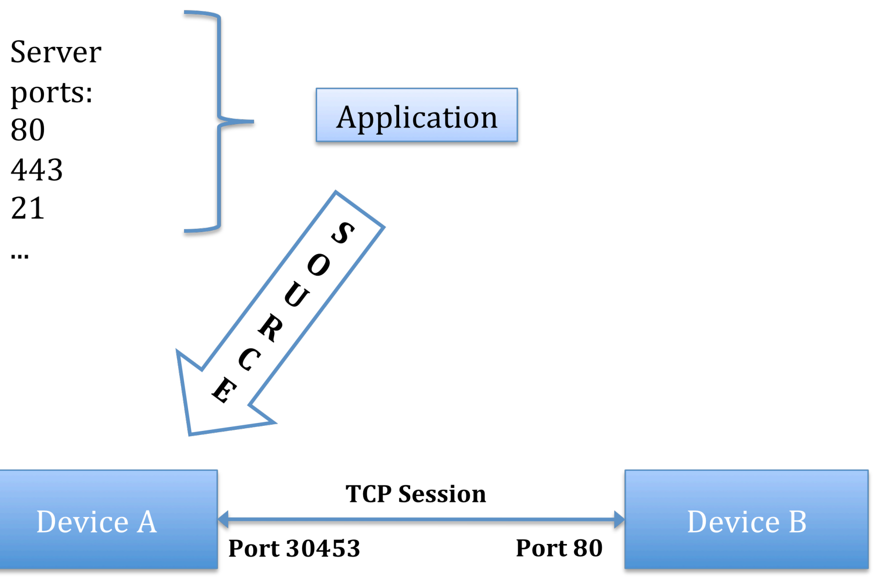

Figure 5. Port Guessing

5.2.2.2 Port validation

Another method used to detect the source of a connection is to perform a test of each port in use. By sending a "synchronize" (SYN) request to both ports used, the application may be able to identify the source of the session. A server will respond to a connection request, a client will not. Therefore, a port that responds affirmatively to a SYN packet is very likely to identify the server, and in turn, the source of the session.

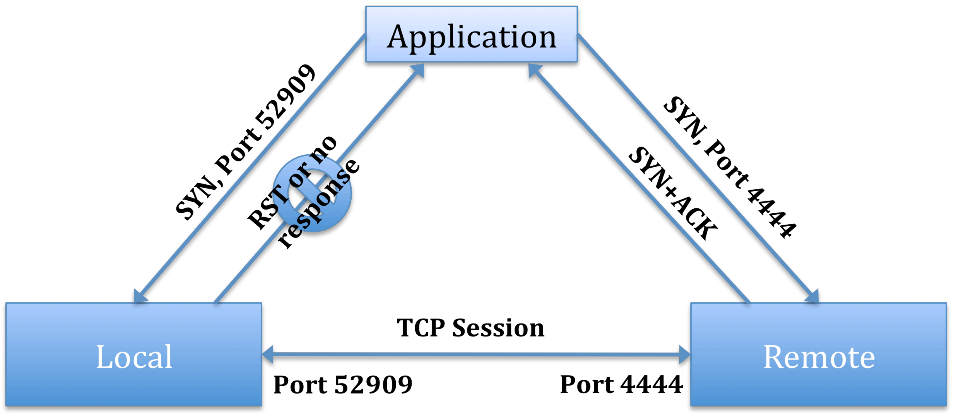

Figure 6. Port Validation

If this method fails, Port Guessing provides a good alternative. Failing both of these methods, the session is marked as "Unknown".

5.3 Remote Device Classification

Once a session is established, it is important that we understand the nature of both devices involved. Local devices are, by definition, known quantities. Remote devices, however, often represent an unknown. If we are to control the communication between Local and Remote, we must have methods of specifying Remote devices effectively. The Internet is a large place; static address lists, while appropriate for some situations, are almost impossible to reliably maintain. In this section, we will examine other methods for classifying Remote devices by identity.

5.3.1 DNS Reverse Lookups

The first method that comes to mind when trying to ascertain the identity of a system is often a DNS reverse lookup. In this case, the IP address is formatted in reverse and added to a special suffix. If the DNS servers of the responsible organization support reverse lookups, the DNS hostname of the device will be returned.

```
$ dig -x 74.125.95.99

; <<>> DiG 9.4.3-P1 <<>> -x 74.125.95.99
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45554
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;99.95.125.74.in-addr.arpa.   IN    PTR

;; ANSWER SECTION:
99.95.125.74.in-addr.arpa. 86399 IN PTR   iw-in-f99.google.com.

;; Query time: 100 msec
;; SERVER: 10.70.10.20#53(10.70.10.20)
;; WHEN: Wed Jun 24 14:32:59 2009
;; MSG SIZE  rcvd: 77
```

Figure 7. Successful Reverse DNS Lookup

Using this method facilitates policy statements that include "domain wildcards". For example, a large number of hosts could be specified by their association with the Trustwave.com domain using the directive:

*. trustwave.com

Statements such as these make it easier to configure policy effectively.

The unfortunate situation with Reverse DNS is that the maintenance of these records is voluntary. Not all Internet hosts will correctly resolve when these queries are issued. Indeed, the author found several legitimate Internet servers during the course of web browsing and instant messaging that did not resolve correctly. This makes Reverse DNS a poor candidate for classification in all but the most controlled environments.

5.3.2 Internet Registry Queries

Where Reverse DNS support is optional, the maintenance of unique Internet addresses is vital. By issuing queries against various Internet Registries using the WHOIS service, addresses can be classified in large groups. The application can expand an address or hostname into a NetRange, and subsequently use that for classification.

A real-world example of this is the process of defining an organization's address range for the purpose of allowing instant messaging. A user connects to AOL Instant Messenger, initiating a long-term session with the address 205.188.211.202. This is one of many AOL Instant Messaging servers. Issuing a WHOIS query for this address returns the following output:

```
OrgName:    America Online, Inc
OrgID:      AMERIC-59
Address:    22080 Pacific Blvd
City:       Sterling
StateProv:  VA
PostalCode: 20166
Country:    US

NetRange:   205.188.0.0 - 205.188.255.255
CIDR:       205.188.0.0/16
NetName:    AOL-DTC
NetHandle:  NET-205-188-0-0-1
Parent:     NET-205-0-0-0-0
NetType:    Direct Assignment
NameServer: DNS-01.NS.AOL.COM
NameServer: DNS-02.NS.AOL.COM
Comment:
RegDate:    1998-04-18
Updated:    1998-04-27

RTechHandle: AOL-NOC-ARIN
RTechName:   America Online, Inc.
RTechPhone:  +1-703-265-4670
RTechEmail:  domains@aol.net
```

Figure 8. WHOIS Query

From examining this output, we can tell that the NetRange for this portion of the organization is 205.188.0.0 – 205.188.255.255. The application can be configured to process this data and map any connections made to this range to a user-defined label, such as "America Online".

This method, while coarse, greatly reduces the amount of administrative overhead associated with classifying remote devices. Care should be taken to ensure that the ranges included truly represent the service portion of the organization; if any portion of the address range has been subleased to an Internet Service Provider, this could lead to an unauthorized connection. That being said, this method closely mirrors the process used by security staff when tracking down an unauthorized session, and is generally accepted as a method of classification.

5.4 Duration

The definition of "long-term session" includes a time element. Depending on the preference of the security staff, this number can vary greatly. A good default is five minutes, though this can be raised or lowered according to network observation. Five minutes is a great deal longer than most valid HTTP sessions, but serves as an aggressive time period for malware and long-term applications.

5.5 Creating Policies

Using the tools outlined above, policy statements can be created to effectively monitor long-term sessions. This section describes common types of policies and offers examples of their use.

The application processes each configuration directive at program launch and builds an internal table with which to enforce policy. For the purposes of this paper, configuration will be presented in simplistic, easily readable form. Actual implementations may use XML, or another more appropriate format to represent configuration data.

5.5.1 Definitions

Groups of devices can be specified using the Classification strategies defined in Section 5.3. This portion of the configuration process assigns labels to groups of devices based on IP address or WHOIS lookup. The following example shows the type of data required by this section:

```
Label : Type : Spec
```

Label is the name assigned to the range.

Type is Remote or Local according to where the device is located.

Spec accepts either static IP address ranges or WHOIS queries. WHOIS queries are placed inside of parentheses to specify them as such. Any NetRange returned from the query will be assigned to this Label.

For example:

```
AOL IM : Remote : (205.188.211.202)
```

Would assign the entire NetRange that includes 205.188.211.202 to the label "AOL IM".

5.5.2 Policies

The goal of the application described in this paper is to alert the security staff to the presence of unauthorized long-term sessions and allow them to take action. This configuration step actually defines what authorized and unauthorized sessions look like. This is accomplished by defining a relationship between Local and Remote resources and then specifying how long a session between these devices should last.

The format used to define these policies is described using the form:

```
Group : Group : Duration
```

Where the first group specified is the Source, the second is the Server, and Duration is specified as a number representing minutes. An asterisk (*) acts as a wildcard value. Wildcards either specify "all Local" or "all Remote" depending on which type of Group is specified. Only one wildcard can be specified. A duration value of 0 specifies that the very existence of the connection should trigger the policy, whereas an asterisk specifies an unlimited duration.

Another configuration section specifies default behavior for each type of device.

```
Default Local : Duration
Default Remote : Duration
```

5.5.2.1 Local User Policies

Local User devices establish a number of long-term sessions in most environments. The most typical application types include:

- Instant Messaging
- File downloads
- Remote control
- Telephony

Policies in this area are typically implemented in a whitelist fashion. The security staff, to determine which servers are contacted, tests authorized applications. In turn, Groups are defined using IP addresses and WHOIS queries to allow long-term sessions with these servers. A policy statement is then created to ignore this type of session.

The following policy triggers on all sessions over 5 minutes made by Local devices, but ignores long-term sessions made from the LAN Group to another Group called AOL IM.

```
Default Local : 5
LAN : AOL IM : *
```

5.5.2.2 Local Server Policies

Due to the fact that servers rarely initiate connections, they are allowed few, if any, valid long-term sessions. Local Server policies will frequently take a whitelist approach.

A simple default policy will trigger when any Server defined as a Local device initiates any connection for over 5 minutes.

```
Default Local : 5
```

To trigger when a Server initiates any connection, a Duration of 0 can be used with a Remote wildcard.

```
Servers : * : 0
```

5.5.2.3 Remote Policies

Remote devices are expected to connect to Local Servers to use resources; therefore duration is an important factor here. As discussed earlier, HTTP sessions are generally short-lived for the purposes of displaying web content. Exceptions include file downloads and Comet applications.

This example specifies 5 minutes as a default for remote connections and creates an exception for the Fileserver group.

```
Default Remote : 5
* : Fileserver : 60
```

Remote connections to User devices is much more uncommon. To detect this behavior, a policy can be added to detect this occurrence.

```
* : LAN : 0
```

6 Conclusion

Like any counter-measure, detection of long-term and inappropriate sessions does not solve every problem. Many attacks against application logic do not look any different at the session level. Malware that does not require a long-term connection may be missed as well, depending on where short-term transactions start and how stringent the policies are. The paper also does not discuss the UDP protocol; tracking UDP "sessions" might be a good area for further research.

While the methods described here provide opportunities to detect and thwart malware, having visibility into another aspect of network behavior is of even greater value. As attacks become more complex, we must pursue greater understanding of how our networks behave. It is not a matter of gathering the data – every stateful firewall employs some of the methods listed here. It is a matter of looking at it the right way, understanding how our endpoints and networks talk to one another, so that we can discern normal behavior from abnormal, and spend our time fixing the right problems.

7 References

RFC 2616, HTTP/1.1 - *http://www.ietf.org/rfc/rfc2616.txt*

Comet: Low Latency Data for the Browser – Alex Russell - *http://alex.dojotoolkit.org/2006/03/comet-low-latency-data-for-the-browser*

WHOIS on Wikipedia - *http://en.wikipedia.org/wiki/Whois*

Transmission Control Protocol on Wikipedia - *http://en.wikipedia.org/wiki/Transmission_Control_Protocol*

Reverse DNS Lookup on Wikipedia - *http://en.wikipedia.org/wiki/Reverse_DNS_lookup*

Metasploit Shellcode Archive - *http://www.metasploit.com/shellcode/*

Know Your Enemy: Tracking Botnets – Honeynet Project - *http://www.honeynet.org/papers/bots*