Graduate Project 3

THE INTRUSION DETECTION ANALYST

Eliora Horst

MANUAL CAPTURES

1	
Diagnosis	Attempted IMAP buffer overload
Wireshark Analysis	All packets appear normal, until packet 9 [Figure 1.1], where the user inputs a long string containing only the character 'A' into the search field, and gives it a narrow range of what to search (Emails 1-4). The attack was unsuccessful, as shown by following the TCP stream [Figure 1.2], which shows the FETCH attempt was able to be completed and returned 0 existing and 0 recent results for the FETCH command.

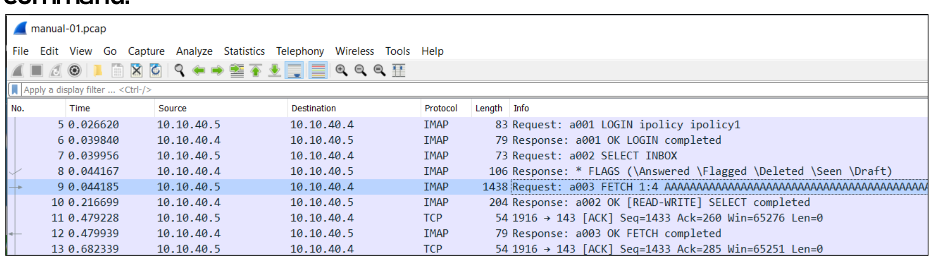


Figure 1.1

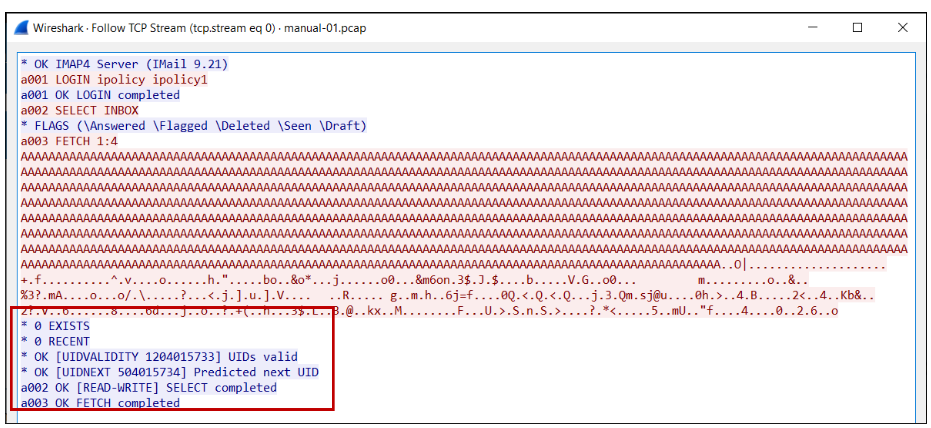


Figure 1.2

<table>
<tr><td>Research and Comments</td><td>There are several possibilities here as to the specific attack, but there are known vulnerabilities with overloading the buffer in IMAP in order to execute a denial of service. CVE-2004-1211 is one option here: "Multiple buffer overflows in the IMAP service in Mercury/32 4.01a allow remote authenticated users to cause a denial of service and possibly execute arbitrary code via long arguments to the … (8) FETCH…"[5] The vulnerabilities in this IMAP configuration can allow for remote validated users to attempt a denial of service attack and to possibly inject code.[4]</td></tr>
<tr><td>Sources</td><td>1. Busylog.net. "Telnet IMAP Commands Note," November 27, 2012. https://busylog.net/telnet-imap-commands-note/
2. Crispin <mrc@cac.washington.edu>, Mark R. "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1." Accessed December 4, 2020. https://tools.ietf.org/html/rfc3501#section-6.4.5
3. "CVE - CVE-2004-1211." Accessed December 4, 2020. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2004-1211
4. FortiGuard "IMAP.Commands.Buffer.Overflow | IPS." Accessed December 4, 2020. https://fortiguard.com/encyclopedia/ips/11599
5. "NVD - CVE-2004-1211." Accessed December 4, 2020. https://nvd.nist.gov/vuln/detail/CVE-2004-1211</td></tr>
</table>

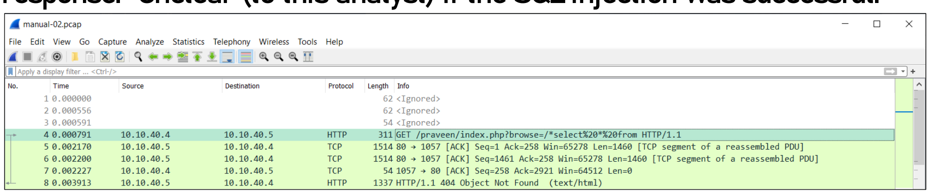
<table>
<tr><th colspan="2">2</th></tr>
<tr><td>Diagnosis</td><td>SQL injection in search bar</td></tr>
<tr><td>Wireshark Analysis</td><td>Basic HTML traffic. Packet 4 [Figure 2.1] of the packet capture shows the attempted SQL injection through the search function of the website, using the SQL phrase "SELECT * FROM." The search then returned a 404 error page in response. Unclear (to this analyst) if the SQL injection was successful.


Figure 2.1</td></tr>
<tr><td>Research and Comments</td><td>SQL injection is an exploitation method used to try and access information contained in a SQL database through unguarded access. Search bars and other input fields on a webpage are easy points of access, as webpages looking to collect some form of data from the user need to have those points of entry open for collection. Users with ill intent can attempt to inject SQL into these fields in hopes of finding a weakness. The user here attempted a basic SQL command and received a 404 error, so going by this packet capture alone, it seems the user was unsuccessful.</td></tr>
<tr><td>Sources</td><td>1. Acunetix. "PHP Injection: Directory Traversal & Code Injection." Accessed December 4, 2020. https://www.acunetix.com/websitesecurity/php-security-2/
2. "SQL Injection in Search Fields | Explore Security." Accessed December 4, 2020. https://www.exploresecurity.com/sql-injection-in-search-fields/
3. w3resource. "SQL Injection Tutorial." Accessed December 4, 2020. https://www.w3resource.com/sql/sql-injection/sql-injection.php</td></tr>
</table>

3

Diagnoses

ICMP Flood Attack

Wireshark Analysis

The capture contains a hair under a million packets. The 'capture file properties' [Figure 3.1] shows that the average PPM (packets per minute) were approximately 15,187. In addition, the capture contains nothing but pings, with requests and replies back and forth. The length of each packet [Figure 3.2] remains identical through the entire capture (verified using the filter: `frame.len!=60 && frame.len!=42`, which yielded no results). This presumably means the payload from each request was identical, and the purpose of which was to overload the network with requests in order to hinder or completely shut down network activity. The source IP indicated as being on the same network, so it is likely that the attacker spoofed their source in order to further befuddle the victim's network.

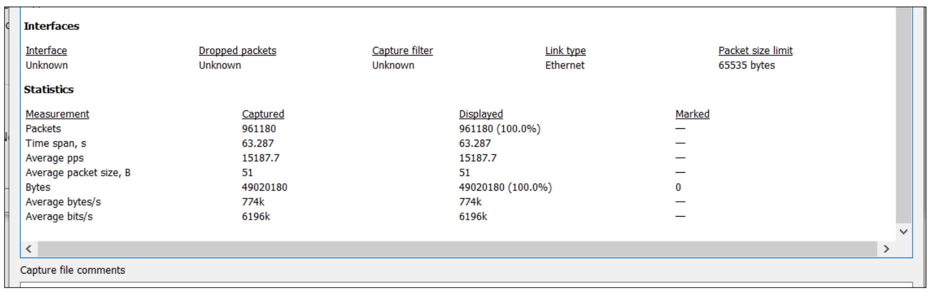


Figure 3.1

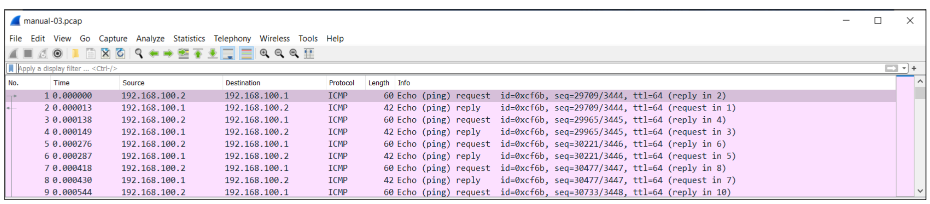


Figure 3.2

Research and Comments

Ping attacks, or ICMP attacks work by sending a literal flood of request packets, as the network is required to send a response to each incoming request. This causes strain on the network, which cannot keep up with the requests, and can lead a complete denial of service. By spoofing the target's IP address, the attacker's own sources are freed to just send packets, while the source is forced to receive and send.[1]

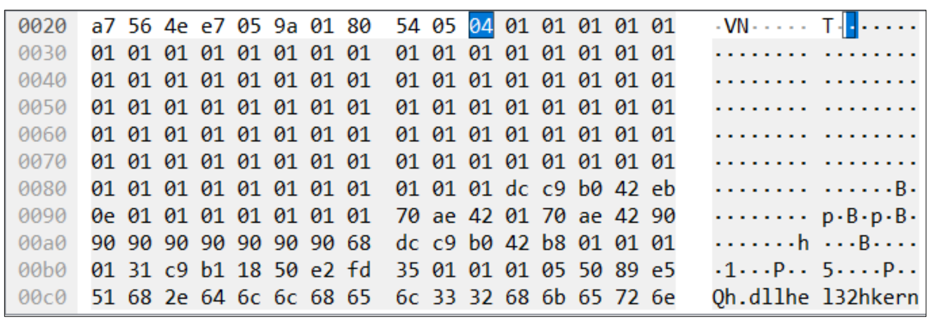
Sources

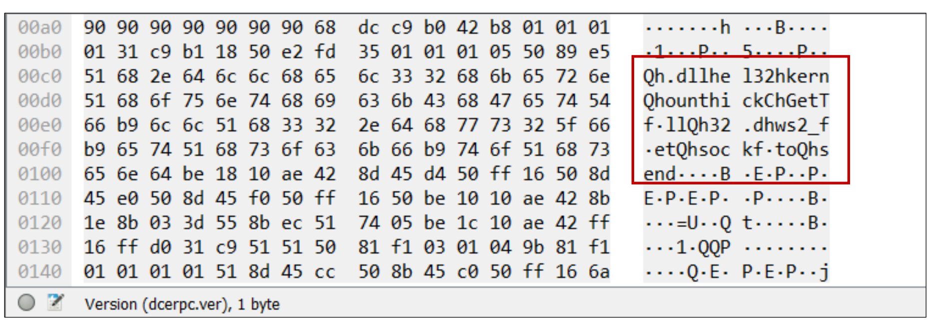
1. "Criscuolo - Distributed Denial of Service – a396999.pdf." Accessed December 4, 2020. https://apps.dtic.mil/dtic/tr/fulltext/u2/a396999.pdf
2. Learning Center. "What Is a Ping Flood | ICMP Flood | DDoS Attack Glossary | Imperva." Accessed December 4, 2020. https://www.imperva.com/learn/ddos/ping-icmp-flood/

SNORT CAPTURES

<table>
<tr><th colspan="2">1</th></tr>
<tr><td>Diagnosis</td><td>MS-SQL Slammer Worm</td></tr>
<tr><td>Snort Output Analysis</td><td>A single packet was analyzed. It was a UDP packet, with no fragmentation, only containing 418 bits. Nothing else of interest. The Slammer Worm uses a single UDP packet to gain entry.

The alert file recognized the packet as “MS-SQL Slammer Worm Activity,” with an additional warning about potentially bad traffic. It also indicated that traffic was going to UDP port 1434, a recognized trait of the Slammer Worm.</td></tr>
<tr><td>Wireshark Analysis</td><td>Identified the packet as using the DCERPC protocol, a protocol used for procedure calls on remote machines, using UDP. Looking at the hex output, there are several features that further confirm the diagnoses. It starts with ‘04’, which indicates to the SQL server that the following data will be name of the online database it is looking. It is then padded with series of ’01.’ [Figure 4.1], another indicator that this is indeed a Slammer Worm. The payload also contains the strings “"h.dllhel32hkernQhounthickChGetTf", "hws2", "Qhsockf" and "toQhsend"[2], which are yet more traits of the Slammer Worm. [Figure 4.2]


0020 a7 56 4e e7 05 9a 01 80 54 05 04 01 01 01 01 01 ·VN····· T·····
0030 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 ········ ········
0040 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 ········ ········
0050 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 ········ ········
0060 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 ········ ········
0070 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 ········ ········
0080 01 01 01 01 01 01 01 01 01 01 01 dc c9 b0 42 eb ········ ······B·
0090 0e 01 01 01 01 01 01 01 70 ae 42 01 70 ae 42 90 ········ p·B·p·B·
00a0 90 90 90 90 90 90 90 68 dc c9 b0 42 b8 01 01 01 ·······h ···B····
00b0 01 31 c9 b1 18 50 e2 fd 35 01 01 01 05 50 89 e5 ·1···P·· 5····P··
00c0 51 68 2e 64 6c 6c 68 65 6c 33 32 68 6b 65 72 6e Qh.dllhe l32hkern
Figure 4.1


00a0 90 90 90 90 90 90 90 68 dc c9 b0 42 b8 01 01 01 ·······h ···B····
00b0 01 31 c9 b1 18 50 e2 fd 35 01 01 01 05 50 89 e5 ·1···P·· 5····P··
00c0 51 68 2e 64 6c 6c 68 65 6c 33 32 68 6b 65 72 6e Qh.dllhe l32hkern
00d0 51 68 6f 75 6e 74 68 69 63 6b 43 68 47 65 74 54 Qhounthi ckChGetT
00e0 66 b9 6c 6c 51 68 33 32 2e 64 68 77 73 32 5f 66 f·llQh32 .dhws2_f
00f0 b9 65 74 51 68 73 6f 63 6b 66 b9 74 6f 51 68 73 ·etQhsoc kf·toQhs
0100 65 6e 64 be 18 10 ae 42 8d 45 d4 50 ff 16 50 8d end····B ·E·P··P·
0110 45 e0 50 8d 45 f0 50 ff 16 50 be 10 10 ae 42 8b E·P·E·P· ·P····B·
0120 1e 8b 03 3d 55 8b ec 51 74 05 be 1c 10 ae 42 ff ···=U··Q t·····B·
0130 16 ff d0 31 c9 51 51 50 81 f1 03 01 04 9b 81 f1 ···1·QQP ········
0140 01 01 01 01 51 8d 45 cc 50 8b 45 c0 50 ff 16 6a ····Q·E· P·E·P··j
Version (dcerpc.ver), 1 byte
Figure 4.2</td></tr>
<tr><td>Research and Comments</td><td>The SQL Slammer Worm works by exploiting the small space where SQL attempts to perform the normal operation of reading in the data. When it encounters the overlong payload, the SQL server rewrites its own stack, unwittingly</td></tr>
</table>

	reprogramming itself with instructions from the worm. The worm then self-propagates by targeting random IP addresses and sending itself to them. This leads to a huge volume of traffic, with the intent to perform a DOS.
Sources	1. "2003 - Attack of Slammer Worm - A Practical Case Study.Pdf." Accessed December 4, 2020. https://www.giac.org/paper/gcih/414/attack-slammer-worm-practical-case-study/103632 2. "2003 - MS SQL SlammerSapphire Worm.Pdf." Accessed December 4, 2020. https://www.giac.org/paper/gsec/3091/ms-sql-slammer-sapphire-worm/105136 3. "What Is DCE RPC?" Accessed December 4, 2020. http://www.doublersolutions.com/docs/dce/OSFdocs/htmls/overview/Dceint63.htm

<table>
<tr><th colspan="2">2</th></tr>
<tr><td>Diagnosis</td><td>Heartbleed / CVE-2014-0160</td></tr>
<tr><td>Snort Output Analysis</td><td>The capture contained 30 packets, and logged two alerts.

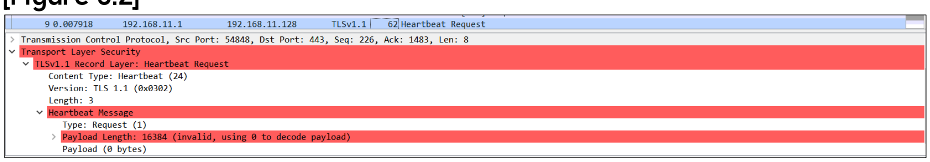
The alert log recognized a "<code>heartbeat read overrun attempt</code>", which is indicative of a Heartbleed attempt. This is the user asking for data. The second alert of "<code>large heartbeat response - possible ssl heartbleed attempt</code>" is also an alert for CVE-2014-0160, where the server attempted to send the request back.</td></tr>
<tr><td>Wireshark Analysis</td><td>The capture opens and closes with a standard exchange of signatures. Packet 9 of the capture contains the heartbeat request – the payload itself is very small, and only contains a few bytes [Figure 5.1]. Packet 26 contains the response, and shows a payload that is considerably higher than the request, at 16,384 bytes. [Figure 5.2]


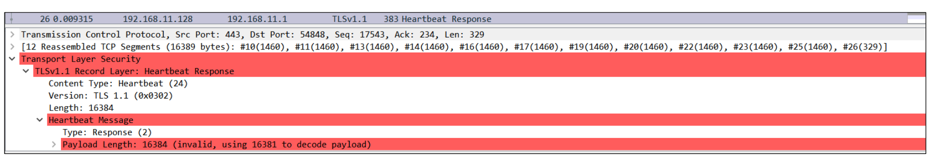
Figure 5.1


Figure 5.2</td></tr>
<tr><td>Research and Comments</td><td>An SSL heartbeat allows two computers to communicate and inform each other that they are still connected. The Heartbleed vulnerability took advantage of the flaws in earlier implementations of OpenSSL, which failed to implement any sort of check to ensure that the incoming heartbeat requests were the length they claimed to be. This allowed attackers to send heartbeat request that claimed to be some small amount but claimed to much larger. The other computer would send a heartbeat back, but since the original message was so small, there was a lot of empty space – so the computer would automatically fill the space with whatever followed the original message in it's memory. By taking advantage of this flaw, attackers could steal up to 64k of memory.</td></tr>
</table>

<table>
<tr><td>Sources</td><td>
1. "CVE - CVE-2014-0160." Accessed December 4, 2020. https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2014-0160
2. Fruhlinger, Josh. "What Is the Heartbleed Bug, How Does It Work and How Was It Fixed?" CSO Online, September 13, 2017. https://www.csoonline.com/article/3223203/what-is-the-heartbleed-bug-how-does-it-work-and-how-was-it-fixed.html
3. "NVD - CVE-2014-0160." Accessed December 4, 2020. https://nvd.nist.gov/vuln/detail/CVE-2014-0160
4. OpenSSL Security Advisory [07 Apr 2014]. Accessed December 4, 2020. https://www.openssl.org/news/secadv/20140407.txt
</td></tr>
</table>

<table>
<tr><th colspan="2">3</th></tr>
<tr><td>Diagnosis</td><td>UDP Port Scanning</td></tr>
<tr><td>Snort Output Analysis</td><td>Output file shows 1000 UDP packets, with one alert, no fragmentation, and 588 UDP sessions created and deleted.

There was a single alert for "UDP filtered portscan."</td></tr>
<tr><td>Wireshark Analysis</td><td>The packet capture shows that all traffic was between the exact same source and destination. About half of the packets are malformed. All the malformed packets also indicate that they used the HART-IP protocol, which is a diagnostic protocol. Every single packet was also the same length. The key here is that the source port never changes, but the destination port is constantly changing [Figure 6.1], a good indicator of scanning activity.
<table>
<tr><th>No.</th><th>Time</th><th>Source</th><th>Destination</th><th>Protocol</th><th>Length</th><th>Info</th></tr>
<tr><td>871</td><td>38.044792</td><td>10.0.32.25</td><td>80.237.98.132</td><td>UDP</td><td>46</td><td>10126 → 29514 Len=14</td></tr>
<tr><td>780</td><td>36.714478</td><td>10.0.32.25</td><td>80.237.98.132</td><td>UDP</td><td>46</td><td>10126 → 29514 Len=14</td></tr>
<tr><td>631</td><td>34.396305</td><td>10.0.32.25</td><td>80.237.98.132</td><td>UDP</td><td>46</td><td>10126 → 9801 Len=14</td></tr>
<tr><td>630</td><td>34.369262</td><td>10.0.32.25</td><td>80.237.98.132</td><td>UDP</td><td>46</td><td>10126 → 48986 Len=14</td></tr>
<tr><td>629</td><td>34.351838</td><td>10.0.32.25</td><td>80.237.98.132</td><td>UDP</td><td>46</td><td>10126 → 20445 Len=14</td></tr>
<tr><td>628</td><td>34.340754</td><td>10.0.32.25</td><td>80.237.98.132</td><td>UDP</td><td>46</td><td>10126 → 31301 Len=14</td></tr>
<tr><td>627</td><td>34.320231</td><td>10.0.32.25</td><td>80.237.98.132</td><td>UDP</td><td>46</td><td>10126 → 43727 Len=14</td></tr>
<tr><td>626</td><td>34.311354</td><td>10.0.32.25</td><td>80.237.98.132</td><td>UDP</td><td>46</td><td>10126 → 14468 Len=14</td></tr>
<tr><td>625</td><td>34.288512</td><td>10.0.32.25</td><td>80.237.98.132</td><td>UDP</td><td>46</td><td>10126 → 52202 Len=14</td></tr>
</table>
Figure 6.1</td></tr>
<tr><td>Research and Comments</td><td>UDP port scans target a specific IP address in order to find open or responsive services. They have internal patterns to randomize which ports they check. Once they receive a response from a port, it means it is open and accessible. This can give attackers access to even further information.</td></tr>
<tr><td>Sources</td><td>
1. Blog, WhatsUp Gold. "Port Scanning 101: What It Is, What It Does and Why Hackers Love It - WhatsUp Gold." Accessed December 4, 2020. https://www.whatsupgold.com/blog/port-scanning-101-what-it-is-what-it-does
2. "Configuring UDP Port Scan Attack Screen - TechLibrary - Juniper Networks." Accessed December 4, 2020. https://www.juniper.net/documentation/en_US/junos-cc19.2/topics/reference/general/security/19.2r1-cc/configuring-udp-port-scan-attack.html
</td></tr>
</table>

<table>
<tr><th colspan="2">4</th></tr>
<tr><td>Diagnosis</td><td>Eternal Blue Trojan</td></tr>
</table>

Snort Output Analysis	The output file registered 577 packets: TCP and IP4 both registered 573 packets, with 4 left over registered as ARP. It also shows in the Preprocessor Statistics that there were 47 total sessions, with 43 aborted, leaving 4. The alert file registered 8 alerts – 4 requests and 4 responses. All were identified as “MALWARE-CNC Win.Trojan.Eternalblue variant echo [either request or response]”
Wireshark Analysis	The alerts in the Snort alert file correspond with 4 SMB echo requests and 4 SMB echo responses. The other SMB packets are establishing the connection to the computer. After initiating a connection, the trojan attempted to send a large NT Transfer packet [packet 14]. Because it is so large, it triggered additional Trans2 Requests. When sending the additional requests, the Trojan deliberately tried to send malformed ones in order to try to exploit vulnerabilities. [e.g. packet 165]. It kept restarting in an attempt to gain entry; 4 times it attempted to request a Trans 2 session setup [packets 12, 257, 270, 565].
Research and Comments	Eternal Blue took advantage of a vulnerability in SMB traffic that allowed attackers to try and disguise the trojan as legitimate traffic being sent back to the computer. The vulnerability becomes available when triggered by malformed Trans2 Requests. Eternal Blue, and its more recent brother WannaCry, are ransomware trojans. It is a trojan specific to Windows, and while Windows released patches and updates to address the issue, users still had to download the updates themselves, and businesses are notorious for running on ancient software.
Sources	1. “NVD - CVE-2017-0144.” Accessed December 4, 2020. https://nvd.nist.gov/vuln/detail/CVE-2017-0144 2. FireEye. “SMB Exploited: WannaCry Use of ‘EternalBlue.’” Accessed December 4, 2020. https://www.fireeye.com/blog/threat-research/2017/05/smb-exploited-wannacry-use-of-eternalblue.html 3. What Is EternalBlue and Why Is the MS17-010 Exploit Still Relevant? “What Is EternalBlue and Why Is the MS17-010 Exploit Still Relevant?” Accessed December 4, 2020. https://www.avast.com/c-eternalblue