Dealing with DoH: Methods to Increase DNS Visibility as DoH Gains Traction

Scott Fether
Dealing with DoH: Methods to Increase DNS Visibility as DoH Gains Traction

GIAC (GCIA) Gold Certification

Author: Scott D. Fether, sfether711@outlook.com
Advisor: Johannes Ullrich

Accepted: April 3, 2020

Abstract

Microsoft is planning to implement DNS over HTTPS (DoH) in the native Windows DNS Client (Jensen, Pashov, & Montenegro, 2019). Firefox and Chrome have already implemented this protocol in their browsers. Because of DoH’s encrypted nature and use of port 443, security analysts will need to adjust their log collection and analysis techniques. Much of the literature available regarding DoH suggests either preventing the use of DoH (Hjelm, 2019, p. 20) or utilizing SSL/TLS proxies to inspect the queries (Middlehurst, 2018). Firefox can generate host logs on DoH resolution, which includes unencrypted queries and answers. This research will explore various inspection and logging techniques that will identify the most effective approach to analyzing DoH.
1. Introduction

DNS over HTTPS (DoH) will soon be available within the native Windows DNS Client (Jensen, Pashov, Montenegro, 2019), and network analysts must identify the most effective log collection and analysis techniques. Some of the current methods of analysis for DoH include either preventing the use of DoH (Hjelm, 2019, p.20) or utilizing SSL/TLS proxies to inspect the queries (Middlehurst, 2018). Internet Service Providers (NCTA, 2019), governments (Gatlan, 2019), and many security professionals (Warran & Buchanan, 2019) have spoken out against DoH due to the challenges it presents to current filtering, identification, and blacklisting methods. Encrypted DNS is beneficial to privacy and security overall, and it is the researcher’s opinion that security professionals must embrace encrypted protocols while developing robust techniques to identify malicious traffic.

Encrypted DNS provides increased privacy, and can prevent man-in-the-middle attacks. It is also an imminent technology that security professionals must face. Malicious actors have always abused DNS in C2 channels and for data exfiltration. The introduction of DoH will only prove to be more enticing to adversaries. Malware such as Godlua and PsiXbot have already made use of the protocol (“New Godlua Backdoor Found Abusing DNS Over HTTPS (DoH) Protocol”). DoH traffic is not easily distinguished from other HTTPS traffic, so it will be miscategorized by tools such as Zeek. Blocking DoH may seem like a viable option, but due to its use of port 443, this becomes difficult, especially if custom malware makes a presence on the network (Cihodariu, 2019).

DoH, in its current implementation, can and should be utilized safely with the addition of host logging, proxy methods, and a Security Event Information Management (SEIM) solution. Firefox can provide logs on DoH resolution, which includes unencrypted queries and answers. These logs can be shipped to a SEIM and visualized, providing a more complete DNS picture (Hjelm, 2019, p.21). As Chrome and Microsoft refine their implementations, it is expected that host and application logs will contain the information that analysts need in an unencrypted format. This research will implement custom Elasticsearch, Logstash, and Kibana configurations to view DoH information on
Security Onion. It will also attempt to measure the effectiveness of proxying traffic destined for port 443.

2. Research Method

To determine the effectiveness of various logging and analysis techniques, a VMWare ESXi lab environment was used to create three different security architectures. All scenarios implemented a pfSense firewall as the gateway for a LAN. Each LAN included one Windows 10 version 1909 virtual machine (VM) with Firefox version 17.0.1 installed. Firefox was configured to utilize DoH by following the procedures outlined at https://support.mozilla.org/en-US/kb/firefox-dns-over-https (“Firefox DNS-over-HTTPS”). Cloudflare was configured as the DoH server in all scenarios. Each scenario also included a standalone Security Onion VM to capture and analyze traffic.

With each of these architectures in place, Firefox browsed several websites. Once at least 100 DoS queries were generated, the researcher checked Security Onion to identify how the traffic was categorized and analyzed. This method revealed to analysts the level of detail each security architecture presented, and whether this provided value to their ability to analyze traffic or intrusions.

Once this baseline was gathered, a tool called GoDoH simulated a malware infection that utilized DoH as a command and control method. GoDoH is a proof of concept that uses DoH as a means to issue OS commands or transfer files. The effectiveness of each scenario was measured to determine which architecture was best suited to detect malware using DoH.

2.1. Scenario 1 - Baseline

Scenario 1 served as the baseline for data collection. No additional logging was enabled on either VM. This scenario represents the security architecture that an enterprise would likely implement without consideration for handling DoH traffic. This basic LAN included the Windows 10 VM with Firefox and Security Onion. Prior to collecting data, the assumption was that DoH would evade Security Onion’s ability to track name resolution due to its encrypted nature and utilization of the standard HTTPS port 443.

Scott D. Fether, sfether711@outlook.com
2.2. **Scenario 2 – Firefox HTTP logs with ELK**

Scenario 2 enabled some enhanced logging on the VMs. This scenario intended to identify how much additional DoH visibility could be gained by enabling Firefox HTTP logging. Firefox HTTP log modules were enabled utilizing the “about:networking” configuration option. Specific steps for enabling these log modules are located at https://developer.mozilla.org/en-US/docs/Mozilla/Debugging/HTTP_logging (“HTTP logging”). In this lab, the logging modules for “timestamp, sync, nsHostResolver:5” were enabled to capture DoH queries and answers at the unencrypted level. The remaining modules were disabled to prevent the log files from growing too large. After logging was enabled, the researcher installed Filebeat on the Windows 10 VM. Filebeat was configured to ship the Firefox logs to Security Onion’s Elasticsearch, Logstash, and Kibana (ELK) instance.

For Security Onion to correctly parse custom Firefox logs, a custom Logstash configuration file must be written. Custom visualizations and dashboards in Kibana were also written so that the data could be appropriately visualized for analysis. The specific configuration files, visualizations, dashboards, and procedures are published on the researcher’s GitHub page (Figure 1) at https://github.com/SigWarrant/Firefox-DoH-SO. The assumption before data collection was that enhanced logging would provide an analyst with unencrypted DoH queries and answers.
2.3. **Scenario 3 – Security Onion with PolarProxy**

Scenario 3 focused on proxying DoH traffic at the network level, allowing Security Onion to analyze the DoH traffic in an unencrypted form. This architecture started with the same setup as Scenario 1. From that baseline, PolarProxy was installed on the Security Onion VM using the instructions provided by NETRESEC at [https://www.netresec.com/?page=Blog&month=2020-01&post=Sniffing-Decrypted-TLS-Traffic-with-Security-Onion](https://www.netresec.com/?page=Blog&month=2020-01&post=Sniffing-Decrypted-TLS-Traffic-with-Security-Onion) (Hjelmvik, 2020). Port-forwarding was configured on the pfSense firewall to redirect all port 443 traffic to the Security Onion where it would be decrypted by PolarProxy. Upon decryption, PolarProxy utilized tcpreplay to feed the unencrypted data to Security Onion for analysis. A graphic depiction of this architecture is shown in Figure 2. In this configuration, all HTTPS traffic (including DoH) was analyzed in its unencrypted form while remaining encrypted during network exposure. Before data collection, the assumption for this scenario was that the unencrypted DoH traffic would be parsed as HTTP for more detailed analysis by Security Onion tools.

Scott D. Fether, sfether711@outlook.com
Dealing with DoH: Methods to Increase DNS Visibility as DoH Gains Traction

Figure 2. Security Onion with PolarProxy (Hjelmvik, 2020)

2.4. Malware Infection

Scenario 1-3 was infected with a simulation of malware utilizing DoH as a command and control function. A C2 tool called GoDoH was used to infect the Windows 10 machines in each scenario. According to Sensepost, “Godoh is a proof of concept Command and Control framework, written in Golang, that uses DNS-over-HTTPS as a transport medium” (Sensepost, 2019). GoDoH uses a combination of DoH and domain fronting to establish a covert C2 channel. Domain fronting is a circumvention technique that obfuscates the domain of an HTTPS connection (“Domain fronting”, 2020). One perspective suggests that:

Using legitimate and more often than not, trusted domains such as google.com to “front” your traffic to a C2 has been a thing for a while. Domain fronting has seen a decent enough uptake where it has been used for censorship circumvention as well as in some malware campaigns (SensePost, 2018).

To establish a GoDoH C2 channel, a public domain name was registered using NameCheap called “dohforthewin.xyz.” An Amazon t2.micro instance was configured as

Scott D. Fether, sfether711@outlook.com
the authoritative name server for this domain. GoDoH was installed on the server and configured to listen for queries on port 53.

On the client, the GoDoH agent was downloaded and configured to run utilizing the preferred DoH server. For consistency, the CloudFlare server was utilized to establish the DoH C2 session. This configuration enabled the C2 server to utilize CloudFlare as a means to forward hidden data within DoH queries and answers. The assumption for this portion of the research was that each solution would render the C2 channel differently. One solution may provide more complete visibility on the trace, making it easier for an analyst to discover the malicious traffic.

3. Findings and Discussion

To identify how the DoH traffic is handled in each scenario, it is essential to understand which DoH servers Firefox is communicating with. By default, Firefox utilizes https://mozilla.cloudflare-dns.com as its server. Using Firefox’s own DoH resolver that is available in the “network:config” page, it is possible to find the IP addresses of this url (Figure 5). This information was utilized to analyze the traffic with these source/destination IP addresses.
Figure 5. IP Lookup of Mozilla’s Cloudflare DoH Server

3.1. Scenario 1 Findings

In this baseline scenario, the DoH traffic was captured by Security Onion, and placed in Zeek’s ssl.log in its encrypted format. This behavior was expected. Zeek was unable to identify the traffic as DNS due to its use of port 443 and encrypted nature. Furthermore, when the captures were viewed in Wireshark there were just a few sessions that appeared to consist of multiple queries and answers. The traffic was using TLS 1.3, and without a proxy, this did not provide the analyst with detailed data. The only way to separate the traffic from other SSL traffic was to filter for the destination IPs of CloudFlare’s DoH servers, which is depicted in Figure 6.
Dealing with DoH: Methods to Increase DNS Visibility as DoH Gains Traction

Scott D. Fether, sfether711@outlook.com
Dealing with DoH: Methods to Increase DNS Visibility as DoH Gains Traction

The same network trace shows as a single SSL session when opened in Network Miner because multiple queries are performed utilizing the same source and destination port. This lack of detailed data validates the assumption that DoH can effectively hide the nature of its traffic unless additional methods are utilized.

Although the nature of the traffic is shrouded in encryption for this scenario, one tool did prove its ability to show the presence of DoH traffic on the network, regardless of the encrypted format. Suricata fired multiple alerts when DoH traffic initialized with the “ET POLICY Observed Cloudflare DNS over HTTPS Domain (cloudflare-dns.com in TLS SNI)” rule. This rule, sid 2027695, utilizes the TLS Servier Name Indication (SNI) to identify standard DoH traffic on the network. The rule reads the content of the packet where the SNI is located and looks for “cloudflare-dns.com.” It is useful in identifying the presence of the protocol on a network, and can effectively be utilized to identify DoH traffic if it is against policy. SGUIL was used to view these Suricata alerts. A screenshot of the findings is shown in Figure 8.

Figure 8. Suricata Alerts for Cloudflare DoH Traffic
3.2. Scenario 2 Findings

This architecture was the same as Scenario 1 from the network capture perspective, but it did attempt to provide visibility on the queries at the host logging level. Host log visibility was accomplished by shipping Firefox HTTP logs to Security Onion via Filebeat. The Security Onion ELK stack provided a centralized place for the analyst to parse and visualize these logs in an unencrypted format. Using the methods outlined in the researcher’s GitHub page https://github.com/SigWarrant/Firefox-DoH-SO, Scenario 2 enabled Security Onion to see queries and answers, aggregate those beats into a visualization, and build a dashboard for a centralized location for the information. Figure 9 shows the dashboard.

![Custom Kibana dashboard for Firefox beats](image)

Using this method, the enhanced logging allowed each query that Firefox performed to be parsed by Logstash as a single event. These events were then sent to Elasticsearch for indexing, and were visualized via Kibana. The DoH queries and answers were aggregable by accessing the “query.keyword” and “answers.keyword” fields within Elasticsearch. This dashboard allows an analyst to see how many times and
Dealing with DoH: Methods to Increase DNS Visibility as DoH Gains Traction

at what frequency a query is performed. It also permits an analyst to view all IP addresses that were returned for a particular query.

Scenario 2 was an improvement over the baseline configuration, because by accessing the host logs in their unencrypted format, a security analyst can view the queries and answers that Firefox is utilizing. At the same time, the queries are not viewable over the network, so the inherent privacy protections that DoH provides remain intact. In that respect, this method is an improvement over Scenario 1; however, this only accounted for Firefox activity. If the host was infected with malware that utilizes DoH as a C2 channel, the logs would not be visible to an analyst. Custom C2 activity would only be accessible from a network capture perspective, and this scenario did not improve upon that situation.

3.3. Scenario 3 Findings

Scenario 3 utilized PolarProxy to decrypt all traffic destined for port 443. PolarProxy sent unencrypted copies of the traffic to Security Onion on port 57012 using tcpreplay. This proxy method presented data to Security Onion in an unencrypted format. The expected behavior was that all the DoH traffic would be parsed by Zeek and placed in the http.log. Upon generating traffic, however, this was not the case consistently. While Zeek categorized some of the traffic as HTTP, other DoH traces were placed in the weird.log for various reasons. Zeek had three reasons for placing the traffic in weird.log: "bad_HTTP_request", "line_terminated_with_single_CR", and "unknown_HTTP_method". A smaller portion of traces did make it into the http.log. In this test, six traces made it into weird.log, and only one made it into http.log. Zeek log placement is shown in Figure 10.
Dealing with DoH: Methods to Increase DNS Visibility as DoH Gains Traction

Scott D. Fether, sfether711@outlook.com

Figure 10. Zeek categorization of unencrypted DoH traffic

Regardless of which log Zeek places the trace in, each one can be analyzed in plain text via Wireshark. In Figure 11, the researcher used the Wireshark filter “dns.flags == 0x8180” to view all standard query responses coming from the Cloudflare server. Queries and answers were seen in plain text. Furthermore, since the traffic was analyzed at the network level, the analyst has complete information as to the data contained in the transmission, including the query, query type, class, time-to-live, data length, and CNAME. If an attacker were to misuse the DoH protocol for C2 or tunneling, the analyst would have all the data necessary to identify the protocol misuse.
Security Onion has more tools available than just Zeek, and this makes it possible to utilize intrusion detection tools like Suricata for automated detection. This was evident in Scenario 1, where DoH traffic to CloudFlare triggered an alert. When analyzing traffic in an unencrypted format, Suricata rules may need to be customized to trigger on decrypted traffic.

As the analyst was building the security architecture for Scenario 3, it was discovered that a tool called NetworkMiner, already included in the Security Onion distro, has a parser for DoH traffic built-in. Erik Hjelmvik demonstrated this at CS3Sthlm 2019 (Hjelmvik, 2020). When traffic is sent to NetworkMiner in its unencrypted form, all DoH traffic can be viewed on its DNS tab. This can be accomplished simply by configuring NetworkMiner to listen on port 57012. The feature is called PCAP-over-IP and is available as an option under the “File” menu.
Optionally, the analyst could open a decrypted pcap in NetworkMiner to view the traffic in the same way. Still, the ability to capture the traffic live using PCAP-over-IP provides additional ways to conduct incident response. If a security analyst suspects that DoH is being utilized maliciously, he or she could redirect HTTPS traffic to Security Onion with PolarProxy, and use this tool as a way to capture all DNS and DoH traffic for that particular machine. NetworkMiner parses the traffic in the same way that Wireshark does but filters DNS specifically all in one tab.

These tools work very well when you are expecting to find DoH traffic on your network, but if DoH were prohibited, the proxied traffic would present a bit of a problem. In Scenario 1, it was observed that Suricata fired multiple alerts concerning the presence of DoH traffic. This rule used TLS SNI to identify traffic going to CloudFlare, and could be very useful if the enterprise’s policy is to prohibit DoH to specific servers. In Scenario
3, all the traffic was decrypted before being sent to Security Onion’s suite of tools. Because of this, no Suricata rules fired, showing the presence of DoH traffic. While analyzing the traffic in its decrypted form proved useful for inspecting the packets in detail, it did render some alerts benign.

Scenario 3 provided the most visibility into DoH traffic out of the three options. One of the benefits is that DoH traffic remained encrypted over the network. The only place that the decrypted traffic could be viewed was on the Security Onion host itself. This scenario accounted for all DoH traffic, whether it originated from Firefox, Chrome, Windows, or custom applications. Some tools, such as Suricata were not as useful as they were in Scenario 1, but it was the most robust scenario for identifying DoH misuse.

### 3.4. Malware Detection Findings

Upon observing the behavior of regular DoH traffic in each scenario, the malware was introduced into each lab environment. A GoDoH C2 server was configured on an Amazon t2.micro instance, and the domain name “dohforthewin.xyz” was registered for the C2 to use. Once the agent was installed and configured to communicate to the C2 server utilizing CloudFlare DoH, commands could be issued to the client utilizing DoH as a covert channel. Figure 14 shows the success of this C2 channel through the issuance of the “systeminfo” command. Several commands were issued in each lab instance to generate enough traffic for Security Onion analysis.
Dealing with DoH: Methods to Increase DNS Visibility as DoH Gains Traction

As soon as the GoDoH agent was initiated in Lab 1, the Suricata alert for Cloudflare DoH traffic fired and was visible in SGUIL. This provided a starting point for the analyst to view the trace. SGUIL gives the user the ability to open the offending network trace by merely right-clicking the alarm and opening it in Wireshark. As expected, the trace contained encrypted TLS traffic, but the data still had the recognizable traits of a C2 channel. When filtering only for frames that included application data, some patterns emerged. These patterns included a C2 beacon interval of 10 seconds and consistently-sized data segments. The alert quickly leads to the conclusion that C2 data exists on the network, as seen in Figure 15.

Scott D. Fether, sfether711@outlook.com
Dealing with DoH: Methods to Increase DNS Visibility as DoH Gains Traction

Scott D. Fether, sfether711@outlook.com

The frames that included deviations from the consistent lengths of 383, 85, and 92 are presumably where data exchange occurred between the C2 server and agent. That data presents itself in a completely encrypted format in both Lab 1 and Lab 2. Lab 2’s host logging produced no further insight for the analyst because the GoDoH agent was running in its process outside of Firefox’s host logging capability.

While the C2 channel was able to be identified in these all scenarios, it could easily be lost in a network where connections to CloudFlare are permitted. It is likely that an analyst would disable the Suricata rule for Cloudflare DoH in order to minimize the number of alerts that are produced. If that were the case, the Suricata rule would never fire. In a restrictive environment, however, this Suricata alert would stand out immediately and require further analysis.

Scenario 3 did not produce a Suricata alert, but this was expected based on previous analysis. The traffic was placed into Zeek’s http.log in a decrypted format. This presented a much different view of the data. Wireshark, nor any other tool, parsed the traffic as DoH, but when viewed, it was quickly identified as DoH traffic by analyzing the HTTP2 headers. Not only was the analyst able to see the 10-second interval between beacons and consistency in data length, but he was able to see the C2 malware domain in plain text. This provides a distinct advantage over Scenario 1 and Scenario 2, because the domain name can be reported as malicious once the channel is identified. A view of C2 beacon traffic is depicted in Figure 16.

Figure 15. GoDoH C2 Beacon in Wireshark

10 Second Interval Beacon
Consistently Sized Data Segments
Figure 16. GoDoH C2 Beacon – Decrypted

The capture in Figure 16 only shows the C2 keepalive traffic but does not represent actual data transfer between the agent and the C2 server. In Lab 1 and 2, the analyst was only able to guess that data was being transferred by observing data length differences in the traffic. Since the traffic was still encrypted by TLS, no other patterns could be seen. In Scenario 3, however, different data patterns can be observed which can identify or create a signature for the C2 channel. Figure 17 shows actual data transfer between the server and the agent. A pattern can be observed within the DNS query where segments are distinguished by number. Presumably, when the C2 server commands the agent to issue “systeminfo” to the OS, that data is encrypted or encoded in segments. Still, an identifier for each segment is given so that the server can reassemble the OS output on the other end. Note that the third “.” delimited field within the dns-query increases by 1 each time new data is sent. This is presumed to be a segment identifier.
While further work would need to be done to read the data that was transferred, it is clear that this proxied traffic provides some additional data to an analyst such as the structure of the queries and the plain text name of the C2 server. The weakness of Scenario 3’s security architecture is that it renders some Suricata rules useless if they were written to trigger on TLS traffic using SNI. Analyzing DoH in its encrypted format as depicted in Scenarios 1 and 2 did have the benefit of utilizing some built-in Suricata rules for identifying DoH to particular service providers; however, Scenario 3 provided the analyst the most detail when further investigation was necessary.

4. Recommendations and Implications

DoH introduces increased privacy and malware protection over the legacy DNS protocol. It also presents a challenge for information security personnel who have relied on DNS as a mechanism to verify compliance of acceptable use and aid in intrusion detection. Just as DoH will be utilized as a way to harden the vulnerability of DNS, it is expected that malicious actors will use it to hide their activity. Information security professionals must go further than attempting to block the protocol by identifying methods to control and analyze DoH.

4.1. Recommendations for Practice

This research tested the effectiveness of three different security architectures when DoH was present on the network. It is clear that current architectures have limited detection capability and do not provide enough detail for malicious DoH traffic to be analyzed appropriately. Scenario 2 of the research showed that application logging could aid information security personnel in viewing DoH queries and answers without compromising the encryption of DoH traffic itself. The additional visibility Scenario 2 provides was limited to the application in which host logging was being performed, however. This practice is recommended in environments where DoH is being utilized, and it is especially useful where proxies are not feasible. This research developed configurations for dealing with Firefox DoH logs. If other applications are being used,
researchers can establish configurations to parse those logs as well. This will result in increased visibility as applications will rely on traditional DNS less in the future.

Scenario 2 did not, however, account for the visibility of DoH traffic that could be introduced by custom malware. The ability to alert on DoH using TLS SNI was already a feature in the baseline Scenario 1 environment. Malware has already been identified that utilizes DoH as a C2 channel. This practice is expected to increase in the future. The ability to alert on DoH by using Suricata rules is useful, but it does not aid further analysis, such as identifying the malicious malware domain name. Scenario 3 accounts for this by proxying all traffic that uses port 443. Research showed that, while Zeek did not correctly categorize the traffic consistently, the traces were readable and available for analysis. Details such as the structure of C2 queries and the C2 domain name were revealed by proxying the traffic. Dealing with DoH traffic in its unencrypted form allows analysts to write additional rules using intrusion detection tools such as Suricata. This method provided the most detailed insight into the nature of DoH traffic when compared with other security architectures.

The researcher recommends that information security personnel utilize a combination of encrypted analysis, proxy solutions, and application logging to gain proper visibility of DoH traffic on the network. It would be feasible to run two Security Onion instances – one that analyzes encrypted traffic, and one that works as a proxy – and ship application logs to one or both to gain the advantages of each scenario. With the strengths and weaknesses depicted in this research, slight modifications to a current security infrastructure could significantly increase DoH visibility. Blocking DoH will not always be feasible, especially as Microsoft intends to release the capability within its native Windows DNS client. Security personnel should begin using these methods now so that they can identify the misuse of this protocol in the future.

4.2. Implications for Future Research

Future research should include complete parsers for DoH traffic in security tools. This research showed that tools such as Zeek were not able to categorize unencrypted DoH traffic in a consistent manner. While some tools such as NetworkMiner were able to parse the traffic effectively, more tools need to follow suit. DoH appears to be the future
of encrypted DNS and will become part of every network. Tools must adjust to the new standard to remain effective at identifying malicious traffic.

Just as the researcher utilized Filebeat to ship Firefox logs to Security Onion, additional work must be done to accomplish similar tasks for other applications. Applications should include robust logging capabilities in order to keep this method as useful as possible. Many environments prohibit proxies or make them difficult, so application logging must continue to be part of the picture within security operations centers.

5. Conclusion

DoH is a more secure way to conduct name resolution; however, it presents some challenges for security personnel who need to control its use. This research introduced a method for analyzing application logs and incorporated them into a security distribution. It also identified the effectiveness of proxying 443 traffic, so analysis can be conducted after encryption is removed. Security analysts must learn how to handle DoH traffic as it will become popular in both applications and malware. Using the methods depicted in this paper, the reader should be able to use these techniques to benefit their organization’s security posture.
References


Scott D. Fether, sfether711@outlook.com
Dealing with DoH: Methods to Increase DNS Visibility as DoH Gains Traction


Scott D. Fether, sfether711@outlook.com
Dealing with DoH: Methods to Increase DNS Visibility as DoH Gains Traction


Scott D. Fether, sfether711@outlook.com
Upcoming SANS Training
Click here to view a list of all SANS Courses

<table>
<thead>
<tr>
<th>SANS OnDemand</th>
<th>OnlineUS</th>
<th>Anytime</th>
<th>Self Paced</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS SelfStudy</td>
<td>Books &amp; MP3s OnlyUS</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
</tbody>
</table>