# A security analysis of paid subscription video-on-demand services for online learning

Sora Lee, Jinwoo Kim, Sangjun Ko, Hyoungshick Kim
*Sungkyunkwan University, Republic of Korea*
{*leesora, jinwookim, kkosazz, hyoung*}*@skku.edu*

***Abstract*—A typical online learning service allows users to watch video lectures in web browsers at any time and any place. In many cases of such services, security solutions (e.g., user authentication and access control) have been deployed to secure access to their premium contents to authorized users only who have paid the subscription fee. In this paper, we demonstrate how security solutions in real-world services can be broken easily. We performed an empirical analysis on the effectiveness of the security solutions deployed in the five popular online learning services using a web proxy to analyze the packets transferred between streaming server and web browser for a streaming service. Our experimental results show that one service out of five was vulnerable to password stealing attacks; three services were vulnerable to URL guessing attacks; and two services were vulnerable to cookie cloning attacks. All the websites tested were vulnerable to at least one attack.**

## 1. Introduction

Due to the advances of Internet technologies, the market of online learning has grown dramatically in recent years. The worldwide market for online education reached about 35.6 billion dollars in 2011 and the annual growth rate is estimated at around 7.6% [1]. Also, online education services are very popular in South Korea; many job seekers and students subscribed paid video-on-demand services to study their interesting lectures at any time and any place. In such a service, a user pays a subscription fee (monthly or annually) to access specifically chosen online lectures.

Since digital contents can be distributed easily over the Internet, security technologies (e.g., user authentication, access control and content encryption) are necessarily deployed in paid video-on-demand services for online learning to protect the media distributed through subscription-based services from unauthorized access. In practice, however, it is not easy to restrict access to digital contents only to a specific set of users. A digital content system appears to be inherently vulnerable to the *analog hole* which refers to an unavoidable vulnerability [2] in copy protection schemes; any copy protected content can be captured and reproduced in an unrestricted form using analog means because the digital content is ultimately converted to a human-perceptible (analog) form (e.g., recording the analog video signals which are produced by a video player). Even with digital formats only, Wang et al. [3] demonstrated that many commercial copy protection systems were insecure against memory analysis. Choi et al. [4] implemented a practical attack to modify the permissions and constraints (e.g., expiration time constraint) in a Digital Rights Management (DRM) protected file. Our study in this paper is more focused on streaming services rather than DRM protected files by exploiting weaknesses in implementations of web applications used for streaming services. The security of web services have already been intensively studied in several applications (e.g., online stores [5] and Java applications [6]). Here, we extended their work by analyzing the security of paid subscription video-on-demand services for online learning.

There are ten major categories of vulnerabilities in web applications identified by the OWASP top 10 project [7]. We particularly investigated the three vulnerabilities of "sensitive data exposure" (closely related to "password stealing attacks"), "missing function level access control" (closely related to "URL guessing attacks") and "broken authentication and session management" (closely related to "cookie cloning attacks"). We will explain those attacks in detail in Section 3. To identify the potential security risks of commercial online learning services, we tested the feasibility of those attacks on the five popular online learning websites in South Korea. We used a web proxy to analyze the packets transferred between streaming server and web browser and identified their security weaknesses. Our experimental results show that one service out of five was vulnerable to password stealing attacks; three services were vulnerable to URL guessing attacks; and two services were vulnerable to cookie cloning attacks. In summary, we make the following contributions:

- We present three important attack methods to illegally access media stream data in video-on-demand services: (1) password stealing attacks, (2) URL guessing attacks and (3) cookie cloning attacks.
- We evaluate the security of real-world video-on-demand services for online learning by testing the feasibility of three attacks on five commercial online learning websites. Our experimental results show that all the websites tested were vulnerable to at least one attack.
- We discuss the best security practices to fix those security vulnerabilities in online learning websites.

The rest of the paper is organized as follows. We briefly explain the security technologies for paid subscription video-on-demand services in Section 2. We present the security analysis framework with the three attacks for testing

in Section 3. In Section 4, we describe the details of our experiments and summarize key results. In Section 5, we discuss general defense mechanisms to mitigate the attacks tested. Finally, our conclusions are in Section 6.

## 2. Background

Video-on-demand is an interactive system that allows a user to watch a video instantly when the user requests the video data. Most video-on-demand services for online learning use a subscription model that requires users to pay a monthly or yearly subscription fee to watch its lecture videos for a specific course. In this section, we describe the general architecture of video-on-demand systems and most common security technologies to protect content owners and service providers against piracy.

### 2.1. Overview of video-on-demand services

Figure 1 shows the high level architecture of video-on-demand systems which consists of three main parties: *video encoder*, *video-on-demand service server*, and *clients*. The video encoder encodes the original video sources into encoded digital stream data and stores the stream data for video-on-demand service server.

Video-on-demand service server has a collection of encoded videos in storage and offers a catalog of the videos to a client. In this architecture, a client represents a device such as PC, mobile phone, or TV to play the requested video. When a client requests the video, the video-on-demand service server transmits the video stream to the client over a streaming protocol such as Real-Time Streaming Protocol (RTSP). For protected contents, user authentication for the client is typically required before the contents to be transmitted.

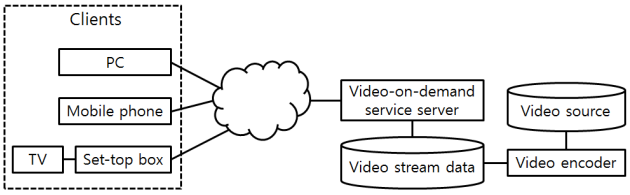


Figure 1. Video-on-demand architecture.

### 2.2. Security technologies for video-on-demand services

We briefly introduce security mechanisms to protect content owners and service providers on video-on-demand services against piracy. The representative mechanisms are as follows: Digital Rights Management (DRM), Real Time Streaming Protocol (RTSP), Real Time Streaming Protocol Secure (RTSPS), session cookie and capability URL.

**DRM.** DRM technologies allow content providers and publishers to control the whole distribution chain and apply flexible usage rules (e.g., watching a video within a predetermined time interval only) [8]. Content providers distribute their digital contents online, but after its distribution they can still exercise control of those contents via DRM. However, many security experts have concerns about the security of DRM technologies. In practice, many commercial DRM systems have proven insecure (e.g., against memory analysis [3] and reverse engineering [4]). To make matters worse, even with a secure DRM implementation, there exists an inevitable problem called the *analog hole* [2] which is the duplication of DRM-protected contents by analog means (e.g., recording the analog video signals which are produced by a video player). DRM is generally applied to content download services rather than streaming services that we are interested in here.

**RTSP and RTSPS.** RTSP (Real Time Streaming Protocol) is a TCP-based streaming protocol designed by RealNetworks, Netscape and Columbia University for low-latency communication of media data such as audio and video [9]. There have been several proposals [10] [11] derived from the basic RTSP to overcome the lack of security features in RTSP which leads to the unauthorized accesses to transmitted data. RTSPS is designed to transmit streaming data over an SSL/TLS connection [12].

**Session cookie.** A session cookie is an identifier which the client receives at the beginning of the session, and then subsequently presents in each request for the duration of the session. Since the holder of a session cookie has a capability to access a website, stealing a cookie may allow an attacker to impersonate a legitimate user. There are a number of standard approaches for constructing secure session cookies. For example, cookie contents should be a cryptographically secure pseudorandom number and be sufficiently long to make guessing a valid session identifier infeasible [13]. Also, session cookies should be invalidated after a timeout period.

**Capability URL.** Capability URLs grant access to a resource (e.g., a media file) to anyone who obtains the URL [14]. Capability URLs are useful because they remove the necessity for users to log in to a website and are easily delegated to others. But this technique can open up some security issues when URLs are exposed to unwanted recipients. Like session cookie, capability URLs must also be hard to guess.

## 3. Methodology

In this section, we present three attack methods to illegally access video stream data in video-on-demand services. We first explain the threat models and then describe those attack methods in detail.

### 3.1. Threat models

In this paper, we consider three different types of adversaries. The common goal of all types of adversaries is

to illegally access video streaming data for a target lecture without paying a subscription fee. The first type of adversary is a *network sniffer*. This adversary is only capable of monitoring the traffic between another user's client application and the web server in the target online learning service. Figure 2 illustrates this type of adversary. This threat model is applied to password stealing attacks.

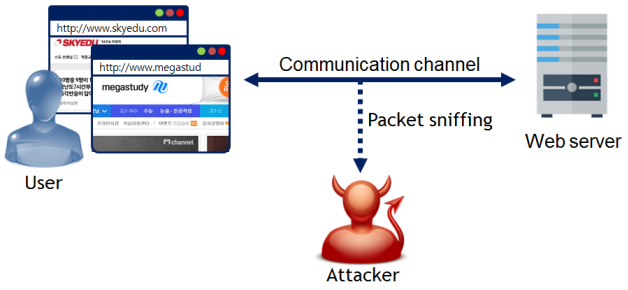


Figure 2. Threat model with a *network sniffer*.

The second type of adversary is a more powerful adversary who acts as a registered user in an online learning website. We call this adversary *malicious user*. This adversary can log in the website and download a client application to directly communicate with the video-on-demand service server. This assumption is reasonable since sample courses and preview lectures are typically available in online learning websites for guest users before paying subscription fees. Moreover, the adversary can pay subscription fees for some courses to analyze the security mechanisms (e.g., authentication methods, encryption methods, message payload structures, etc.) used in the target online learning service. Therefore, the adversary is capable of not only monitoring the traffic between a client and the target web server through a traffic monitoring tool, but also sending modified messages to the web server through a web proxy because the client application is under the full control of the adversary. Figure 3 illustrates this type of adversary. This threat model is applied to URL guessing attacks.

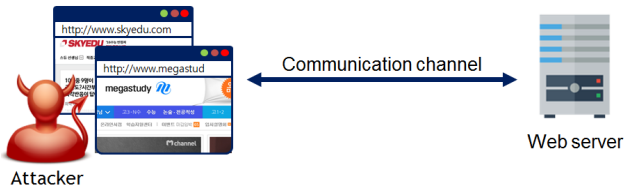


Figure 3. Threat model with a *malicious user*.

The third type of adversary is the most powerful adversary who can access subscribed user's cookies stored in the user's web browser. This type of adversary could be possible in many practical situations. For example, cookies can be stolen by using cross-site scripting attacks (XSS) [15] or malware installed in a victim's device. Also, a malicious user can often steal another user's cookie from a public computer. We call this adversary *malware*. This adversary is capable of stealing a victim user's session cookie from her device. The adversary can impersonate the victim user with the stolen cookie. Figure 4 illustrates this type of adversary. This threat model is applied to cookie cloning attacks.

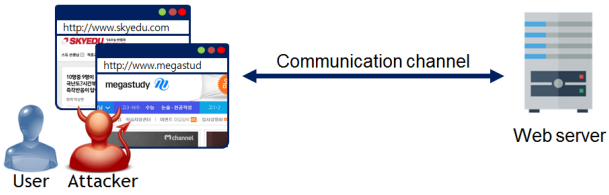


Figure 4. Threat model with *malware*.

## 3.2. Three attacks tested

In this paper, we have particularly focused on the three following attacks which are directly related to unauthorized access to resources in a website by stealing a victim's credential or capability.

**3.2.1. Password stealing attack.** We first try to directly steal login credentials (i.e., user password) by capturing network packets to log in an online learning website. The adversary can collect a victim's credential by analyzing the text fields associated with the password (e.g., `input type="hidden"`) when the communication channel between a client application and the web server is not encrypted properly. For experiments, we use a packet sniffing tool to check whether user password is exposed in plaintext in the view of a *network sniffer* (see Figure 2).

**3.2.2. URL guessing attack.** As mentioned in Section 2, capability URLs grant access to resources based solely on knowledge of the URL. Therefore, if an attacker knows the capability URL of a media stream file, the attack can fully access the media stream file. To achieve this, the attacker can perform the URL guessing attack.

The URL guessing attack is to make many attempts to correctly guess the capability URL of the target media stream file. In theory, capability URLs seem secure enough against guessing attacks because they can be constructed as a long and random string. However, capability URLs are far from being random in many real services. Actually, they often have a well-structured format consisting of the host server, lecture identifier, lecture sequence number and so on. We are interested in guessing a capability URL with a reasonable number of guessing attempts.

For experiments, we analyze capability URLs of media stream files (e.g., provided by preview lectures) for the target lecture and try to guess unknown parts of the capability URLs in a brute force manner in the view of a *malicious user* (see Figure 3).

**3.2.3. Cookie cloning attack.** Since a session cookie is used for user authentication without logging in, stealing a session cookie may lead to session hijacking allowing an attacker to impersonate a victim. Since cookie cloning attacks and defenses (e.g., binding session cookies to IP addresses and cookie expiration time) are well known, we are interested in the possibility of cookie cloning attacks on real websites. For experiments, we create two user accounts on the target online learning website and check whether a user's session cookie can also be used for other user's cookie in the view of *malware* (see Figure 4).

## 4. Experiments

We conducted three attacks explained in Section 3 against the five popular online learning websites in South Korea (hereinafter referred to as Website 1, 2, 3, 4 and 5). According to a website raking service called `SimilarWeb` [16], Website 1 were visited more than 1 million times in the last six months; Website 2, 3, 4 and 5 were visited more than 52 thousands times, 429 thousands times, 859 thousands times, and 297 thousands times during the same period, respectively.

For experiments, we used a PC (with a 2.70GHz Intel Core i5 CPU and 8GB RAM) running the Windows 10 operating system, and equipped with a non-congested 100Mbps LAN that was connected to the Internet.

### 4.1. Security mechanisms of the websites tested

We first analyzed what security mechanisms are actually used in the websites tested. Surprisingly, all the websites tested transmitted media stream data over insecure network protocols such as HTTP and RTSP [9]. This implies that the transmitted streaming media data were basically vulnerable to interception and theft at the network level.

However, all the websites provide some security solutions at the host level. They used proprietary steaming viewer applications that are embedded as ActiveX controls in web pages rather than more-widely accessible streaming software. Those applications were designed to prevent recording by forcibly terminating external recording programs while watching a lecture video on the proprietary viewer. Also, those applications do not work in a virtual machine by detecting if the viewer application itself is running in a virtual machine.

### 4.2. Implementation

**4.2.1. Password stealing attack.** Using a network packet sniffing tool called Wireshark [17], we captured the packets related to user login and analyzed them. From the captured packets, we checked whether the password used in user login is included in plaintext. In this experiment, we successfully obtained the plaintext password from the packets for Website 5 alone (see Figure 5). For all the remaining websites, we failed to obtain user passwords because they were encrypted.

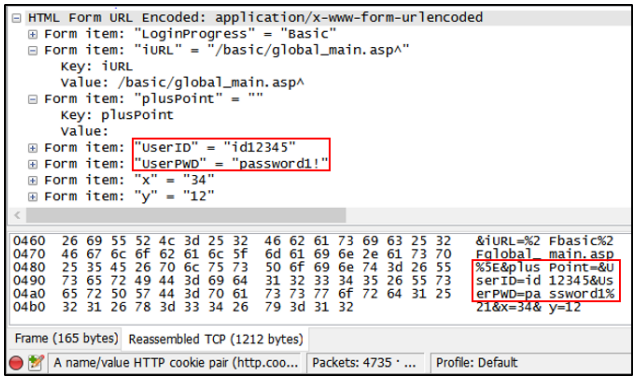


Figure 5. Unencrypted network packet.

**4.2.2. URL guessing attack.** For URL guessing attacks, we first need to know the format of capability URLs. For doing this, we use a web proxy tool called Paros [18] to extract the capability URLs of online lectures at the HTTP level because capability URLs (even for the case of preview lectures) are generally invisible to users for security purpose. Figure 6 shows an example of the capability URL extraction. From this figure, we can see that the URL of a preview lecture video stream was included in the `Referer` header field.

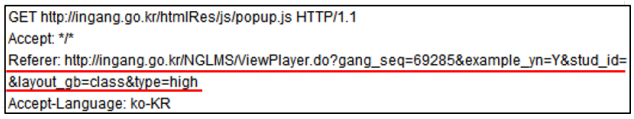


Figure 6. Example of URL extraction (by Paros).

In this experiment, we successfully obtained the meaningful format of capability URLs from the three online learning websites only. We failed to find a general format of capability URLs in Website 4 and 5. In Website 4, capability URLs were independently generated from each other. In Website 5, capability URLs were encrypted at the HTTP level.

After carefully analyzing the extracted URLs from Website 1, 2 and 3, we found how the parameters ("`lectID`" in Website 1, "`gang_seq`" in Website 2 and "`leccd`" in Website 3) included in a URL were changed with lectures and realized that their spaces were not large enough against guessing attacks.

We sequentially generated those parameter values to find the correct values of the target lecture in a brute force manner. Figure 7 shows an example of our implementations. From this figure, we can see that the parameter values for `gang_seq` were sequentially generated.

Finally, we need to check whether a generated URL is valid or not. This test can be implemented as an automatic decision procedure by sending the request message with the generated URL and checking the status of the reply message. If the generated URL is not valid, the error message is returned with the content length which is less than 300

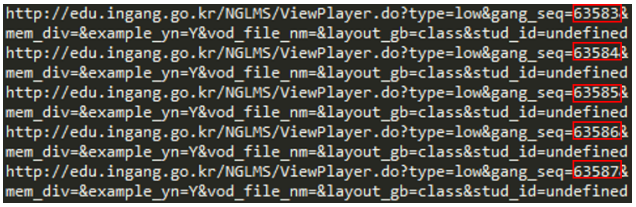


Figure 7. Example of URL guessing attacks.

bytes. With this property, we can automatically test whether generated URLs are valid.

The efficiency of our guessing attacks was highly efficient than our expectation. For Website 1, it took 0.77 seconds on average to obtain a capability URL; 0.20 seconds for Website 2; and 15.41 seconds for Website 3. Their standard deviations were 17.80, 0.32 and 279.38 seconds, respectively. In fact, lecture identifiers were sequentially enumerated in Website 1 and 2.

**4.2.3. Cookie cloning attack.** Again, we use Paros to obtain the cookie information used in websites. Figure 8 shows an example of the cookie.

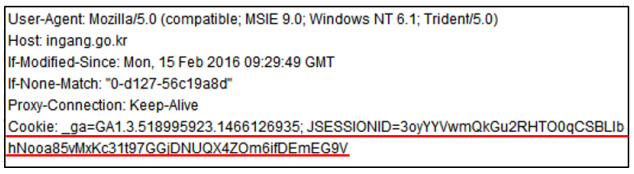


Figure 8. Example of Cookie extraction (by Paros).

In this experiment, we successfully obtained session cookies from Website 3 and 4. Website 1 and 2 did not support the session cookie feature. In Website 5, we cannot use Paros to analyze HTTP packets because they were fully end-to-end encrypted.

For Website 4, when we performed a cookie cloning attack with a stolen session cookie, the web server checked whether the cookie was sent from a valid user (e.g., the cookie owner's device). Therefore, a simple copy of the cookie string might not be working. However, we found that this checking result is finally processed at the client side rather than the server side. Therefore, we can easily bypass this restriction by modifying the server's response with a web proxy such as Paros. Figure 9 shows the JavaScript code to process the server's response. From this code, we can see that the media stream can successfully be played if we set 0 to the `data` variable.

## 4.3. Summary of results

We checked the feasibility of three attacks (described in Section 3) on five commercial online learning websites. Our experiment results are summarized in Table 1. Our experimental results demonstrate that Website 5 was vulnerable to password stealing attacks; Website 1, 2 and 3 were


Figure 9. JavaScript code used in Website 4.

vulnerable to URL guessing attacks; and Website 3 and 4 were vulnerable to cookie cloning attacks. Interestingly, all the websites tested were vulnerable to at least one attack.

TABLE 1. EXPERIMENTAL RESULTS FOR THE FIVE ONLINE LEARNING WEBSITES (O:SUCCESS, X:FAILURE, —:NOT SUPPORTED)

| Service Provider | Attack method | | |
|---|---|---|---|
| | Password stealing | URL guessing | Cookie cloning |
| Website 1 | X | O | — |
| Website 2 | X | O | — |
| Website 3 | X | O | O |
| Website 4 | X | X | O |
| Website 5 | O | X | X |

# 5. Countermeasures

We recommend several defense strategies to mitigate password stealing, URL guessing and cookie cloning attacks described in Section 3.2.

## 5.1. Encryption of web traffic

In order to reduce the risk of password stealing attack, the most straightforward solution is to use a security protocol. Since all web browsers have already supported SSL/TLS [12] which is the de facto standard for encrypting Web-based information interchanges, we might update the systems without incurring significant additional cost.

## 5.2. Use of the best practices for capability URLs

Due to the low entropy of capability URLs used in the online learning websites tested, three out of five those websites were vulnerable to URL guessing

attack. We found that capability URLs in the vulnerable websites were basically generated like `http://[domain name]/[path]/[content ID]` where `[content ID]` is merely a sequentially increasing integer that can be easily guessed.

To avoid the capability URL theft by guessing, `[content ID]` should be randomly chosen and sufficiently long. W3C recommends that good capability URLs include an unguessable unique identifier such as a universally unique identifier (UUID) [19].

### 5.3. Expiration of capability URLs and cookies

To prevents reuse of capability URLs and/or authentication cookies, they have a finite lifetime. For example, it may be suitable to have a capability URL that can only be accessed once, or be expired after a day. This security practice could be helpful to mitigate the damage even when valid capability URLs or cookies are stolen.

### 5.4. Preventing suspicious URL requests

We can prevent suspicious URL requests for guessing valid capability URLs, which may have a pattern that is significantly different from normal users' URL requests. For example, consecutive URL requests with invalid URLs are periodically made during a relatively short time. To prevent URL guessing attempts, a straightforward solution is to limit the number of consecutive failed attempts from a particular device, e.g., based on its IP or MAC address. For example, if we set this number to five, an attacker needs to correctly guess valid URLs within five attempts. If the number of consecutive failed attempts from a device is greater than five, further URL requests from the same device could be ignored or prohibited.

## 6. Conclusion

We analyzed the security of video-on-demand mechanisms on five commercial online learning websites using a web proxy and found all the websites were vulnerable to at least one attack. For example, three websites out of five were vulnerable to URL guessing attacks; a valid URL of a media stream can be obtained automatically (0.77, 0.20 and 15.41 seconds on average were taken to obtain a valid URL, respectively, for Website 1, 2 and 3).

However, our current results are not enough to evaluate the overall security of online learning websites because only five websites were tested. As an extension to this paper, we need to consider performing our tests on a large sample of websites providing streaming services. Moreover, we only considered the three attacks of "URL guessing attacks", "cookie cloning attacks" and "password stealing attacks". Therefore, as another extension to this work, we also plan to test other web security issues such as SQL injection and cross-site scripting attacks.

## Acknowledgments

This work was supported by the NRF (National Research Foundation of Korea) grant funded by the Korea government (No. 2014R1A1A1003707), the IITP (Institute for Information & communications Technology Promotion) grant funded by the MSIP (Ministry of Science, ICT and Future Planning) (No.R-20160222-002755), and the KISA (Korea Internet & Security Agency) grant funded by the MSIP (H2101-16-1001). This research was also supported by the MSIP under the ICT R&D program (R0166-15-1041) and the ITRC (Information Technology Research Center) support program (IITP-2016-R0992-16-1006) supervised by the IITP.

Authors would like to thank all the anonymous reviewers for their valuable feedback.

## References

[1] V. Piccioli, "E-Learning Market Trends & Forecast 2014-2016 Report," tech. rep., Docebo, 2014.

[2] M. Stamp, *Information security: principles and practice*. John Wiley & Sons, 2011.

[3] R. Wang, Y. Shoshitaishvili, C. Kruegel, and G. Vigna, "Steal This Movie: Automatically Bypassing DRM Protection in Streaming Media Services," in *Proceedings of the 22nd USENIX Security Symposium*, 2013.

[4] J. Choi, W. Aiken, J. Ryoo, and H. Kim, "Bypassing the Integrity Checking of Rights Objects in OMA DRM: A Case Study with the MelOn Music Service," in *Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication*, 2016.

[5] R. Wang, S. Chen, X. Wang, and S. Qadeer, "How to Shop for Free Online–Security Analysis of Cashier-as-a-Service Based Web Stores," in *Proceedings of the 32nd IEEE Symposium on Security and Privacy*, 2011.

[6] V. B. Livshits and M. S. Lam, "Finding Security Vulnerabilities in Java Applications with Static Analysis," in *Proceedings of the 14th USENIX Security Symposium*, 2005.

[7] "Owasp top 10 2013." https://www.owasp.org/index.php/Top_10_2013-Top_10. Accessed: 2016-06-29.

[8] F. Hartung and F. Ramme, "Digital rights management and watermarking of multimedia content for m-commerce applications," *Communications Magazine, IEEE*, vol. 38, no. 11, pp. 78–84, 2000.

[9] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," tech. rep., 1998.

[10] J. C. L. Yeung, S. F. and D. K. Yau., "Secure real-time streaming protocol (rtsp) for hierarchical proxy caching," *International Journal of Network Security*, vol. 7, no. 3, pp. 1–20, 2007.

[11] J. R. S. Gruber and A. Basso, "Protocol considerations for a prefix-caching proxy for multimedia streams," in *Proceedings of the 9th International World Wide Web Conference*, 2000.

[12] T. Dierks and C. Allen, "The TLS Protocol Version 1.0," 1999.

[13] S. J. Murdoch, "Hardened Stateless Session Cookies," in *Proceedings of the 16th International Conference on Security Protocols*, 2008.

[14] "Good practices for capability urls." http://w3ctag.github.io/capability-urls/.

[15] D. Endler, "The evolution of cross site scripting attacks," tech. rep., iDEFENSE Labs, 2002.

[16] "SimilarWeb." https://www.similarweb.com/.

[17] "WireShark." https://www.wireshark.org/.

[18] "Paros." https://sourceforge.net/projects/paros/.

[19] J. Tennison, "Good Practices for Capability URLs," 2014.