International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Application Research and Penetration Testing on WebSocket Technology

Priyali Mishra[1], J. Jeysree[2]

[1]Information security and cyber forensics, Masters in Technology, SRM University, Chennai, India

[2]Assistant Professor, Department of Information Technology, SRM University, Chennai, India

Abstract: ***The WebSocket technology provides a bi-directional real time communications for web applications and services. This paper provides the study of the websocket protocol and describes the advantages they provide. This paper concerns about the security related to websockets, discuss the possible solutions and provide the practices for the deployment of the Websocket service. Also it proposes certain security frames that should be implemented in web browsers to ensure the security and privacy of the users. The WebSocket technology is not yet standardized but it is expected that usage of WebSockets will increase significantly in the near future. Overall, WebSockets solve connectivity problems, not the security problems. Several open issues exist, but with a good design and proper implementation of web browsers and web services, the risk level can be mitigated.***

Keywords: websockets, html5, browser security, websocket pentesting

1. Introduction

WebSockets enable browsers to establish a persistent, full-duplex, bi-directional and asynchronous connection to a server. WebSocket API is oftendenoted as HTML5 WebSocket API, although it is actually separated from the HTML5 specification and is currently being developed and specified independently. TheWebSocket protocol is a network protocol running on TCP. It is designed to be implemented in web browsers and web servers but it can be used for other purposes as well.TheWebSocket protocol is independent of Hypertext Transfer Protocol (HTTP), although the protocols have similarities such as using TCP port 80 and the handshake process when initializing a connection. The communication channel can be protected against eavesdropping with TLS, much like HTTPS. The default ports are 80 or 443, such that enterprises are not required to open additional ports in their firewalls.

WebSockets may in turn influence the security of the web in the next few years. WebSocket is still a relatively new concept and not much public research has been done in the field of websocket security. This paper focuses on the possible security concerns in the WebSocket protocol and also on the security features the technology provides.

This paper is structured as follows. The second chapter reviews the background of WebSocket technology. The third chapter focuses on the technical security issues of WebSockets. The fourth chapter includes penetration testing on WebSockets. The final chapter concludes the whole paper and the future work.

2. Background

First we will look at WebSockets: how they are standardized, their properties and intended usage. Section 2.1 discusses the new features and advantages of the technology compared to traditional web communications. Then I want to give some basic idea about penetration testing. Section 2.2 illustrates the usage of the WebSocket.

2.1Features and Advantages of WebSockets

The main difference in WebSockets – compared to the usual network traffic over HTTP – is that the WebSocket protocol does not follow the traditional request-response convention. Once a client and a server have opened a Web-Socket connection, both endpoints may asynchronously send data to each other. The connection remains open and active as long as either the client or the server closes the connection. Web Socket uses single socket to push and pull information between the browser and the server in fullduplex communication. At present, the HTML5 WebSocket is the principal mechanism for promoting the web full-duplex real-time communication.

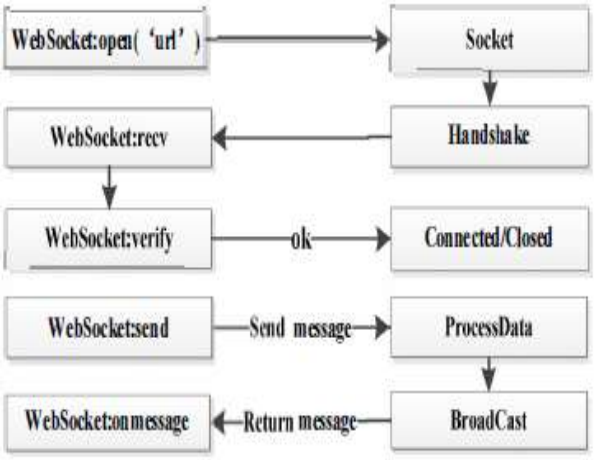

Figure 1: WebSocket connection establishment model

Fig 1 is a WebSocket connection establishment model, WebSocket protocol is essentially a TCP. To establish a Web Socket connection, the client and server upgrade from the HTTP protocol to the Web Socket protocol during their initial handshake, once established, Web Socket data frames can be sent back and forth between the client and the server in full-duplex mode, and this connection will continue to exist until the client side or server side close the connection initiatively. The Web Socket API defines mainly four

callback methods: onopen, onmessage, onclose and onerror to deal with event trigger during the WebSoceket connection.

In contrast, the traditional approach relies on polling. That is, a client opens a new TCP connection and makes an HTTP request to receive data from a server. This requires several round-trips between the client and the server before any actual information is sent. Moreover, a new request is needed by the client for every separate data transmission. Hence, compared toWebSockets, the traditional HTTP request-response convention results in high latency and high amount of network traffic.

2.2 Functionality of WebSockets

A client establishes a WebSocket channel by sending a usual HTTP request to a server and asking for an upgrade of an existing connection. Once it gets upgraded, the subsequent messages are sent using the WebSocket protocol. The WebSocket API can be used in web applications with JavaScript in a quite similar manner to XmlHttpRequests, as listing 1 illustrates.

```
var socket = new WebSocket
  ("ws://<address>:<port>");
socket.onopen = function(e) {
  // socket opened
};
socket.onmessage = function(e) {
  alert("data sent by server: " + e.data);
};
socket.onclose = function(e) {
  // socket closed
};
```

Listing 1: Initializing a WebSocket connection using JavaScript API

Once the connection is active, we can send data to the server or close the connection. This is illustrated in listing 2.

```
socket.send("hello world");
socket.close();
```

Listing 2: Sending and closing the WebSocket connection

3. Security Concerns of Websockets

Every system is just as strong as its weakest link. This applies to security too. If a black hat is able to exploit just one vulnerability he/she may gain access to the whole system. The websocket protocol differs from http in various ways, the differences may also affect to the security of the web, even though that is not always obvious.

3.1Origin Policy

The WebSocket API lets the web application open a connection and send arbitrary data to any server. In contrast to HTTP, the WebSocket protocol uses a verified-origin mechanism. This mechanism lets the target server decide, from which origins it allows connections.

3.2 Firewalls and Proxies

The handshake process is compatible with http and the standard http port 80 is commonly used with the websocket protocol.

In November 2010, a vulnerability related to the websocket protocol has been reported. Malicious usage of a websocket channel allowed cache poisoning of some transparent web proxies. If you use a firewall that does not speak websockets, it could easily, be bypassed using this new communication channel. As a result it should be aware of Websockets. While browser vendors kept their early support for Websockets up-to-date, there are a lot of server implementations that still support older draft versions that may contain vulnerabilities. Using the latest, standardized version of websockets is highly recommended.

3.3 Malicious services

A malicious website can easily set up a remote shell access to the victim's web browser. Once the attacker is able to run arbitrary JavaScript code inside the web browser, she is also able to initiate a WebSocket connection to an arbitrary service. After this, the attacker can utilize the existing WebSocket channel to control the web browser in real-time within the limits of JavaScript. The WebSocket API does not only allow requests to an arbitrary host, but also to an arbitrary TCP port. Thus, the technology can be utilized for port scanning and network mapping in the internal network to which the victim is connected. For instance, if the remote server is listening a certain TCP port, a WebSocket connection attempt to that port initiates a TCP handshake. Since TCP handshake requires couple of round-trips, web application may be able to distinguish open and blocked ports on the remote server according to the response time. Hence, the attacker may be able to bypass the firewalls by setting the victim's web browser to work as a WebSocket proxy between the attacker and the internal network.

3.4 XSS Vulnerability

Cross-site scripting (XSS) is a common type of security vulnerability in web applications. The XSS vulnerabilities allow the attacker to inject malicious code in websites that display parameters or variables that can be modified by user. However, on the websites that utilize WebSockets, the XSS vulnerabilities open up several new threats. For instance, with an XSS vulnerability the attacker may be able to override the callback functions of a WebSocket connection with custom ones. This approach allows the attacker to sniff the traffic, manipulate the data, or implement a man-in the-middle attack against WebSocket connections. In addition, by utilizing an XSS vulnerability, the attacker is able to implement practically any attack.

3.5 Encryption and Data Validation

The WebSocket protocol does not specify mechanisms for authentication or encryption of connections. The protocol suggests using of TLS to provide confidential communication channel. Authentication can be also

implemented using TLS methods for web authentication, such as cookies or HTTP

Authentication: Another relevant point is data validation. The WebSocket protocol specifies that both a client and a server must validate the data received from each other. If invalid data is received, theWebSocket connection should be closed.

4. Penetration Testing on Websockets

With Websockets vulnerability detection is not as easy as with http. When you send a http request there is always a response, where you can easily detect changed outputs, or slow response times etc. With Websockets, communication is bi-directional and asynchronous. As a result automated detection is fairly hard and has to be done manually. The approach to take is similar to stored XSS attacks, where there is often no indication of vulnerability. While the attack payload is stored in a database, it may appear on any page of the web application. While the content may vary, there will always be field values that are user-provided. In order to find vulnerabilities you can enumerate such values. The payloads in use are sometimes called fuzz strings. With automation you are able to test huge amounts of strings. The goal is to cause the web application to behave anomalous.

5. Conclusion

As we have seen in this paper, theWebSocket technology increases the opportunities of real-time web applications. The number of WebSocket endpoints, i.e. servers and clients, will rise in the near future due to the standardization progress. Once theWebSocket technology is widely adopted, the security threats will become more concrete. Whereas theWebSocket services can be deployed safely, they also can be utilized for malicious purposes. Theweb browsers and service developers, should pay attention to the known security issues.

References

[1] I. Hickson, “The WebSocket API,” 2010. [Online]. Available: http://dev.w3.org/html5/websockets/
[2] ”http://dl.packetstormsecurity.net/papers/attack/AC07815487.pdf”
[3] Internet Engineering Task Force (IETF). The Web-Socket Protocol, RFC 6455.
[4] M. Schema, S. Shekyan, and V. Toukharian. Hacking with WebSockets. BlackHat USA 2012.
[5] World Wide Web Consortium W3C. The WebSocket API: W3C Candidate Recommendation. “http://www.w3.org/TR/websockets/”. [Online; received 25 Sep 2012].