Burp Suite training tutorial: Part 3 – Sequencer, decoder and composer

Karthik R, Contributor

Read the original story on SearchSecurity.in.

>>For more on the Burp Repeater and Intruder tools, refer to the second tutorial in this series<<

In the two earlier installments of our Burp Suite training tutorial, we covered several tools available in Burp Suite, including Burp proxy, Burp intruder, Burp spider and Burp repeater. We have also seen how to test a Web application for XSS vulnerabilities and SQL injection vulnerabilities. In this final installment of the Burp Suite training tutorial, we shall cover three more tools of Burp Suite: *sequencer*, *decoder* and *comparer*.

Burp sequencer

The Burp sequencer tool is used to check for the extent of randomness in the session tokens generated by the Web application. Brute force attacks enumerate every possible combination for gaining authentication from the Web application. Thus it is important to have a high degree of randomness in the session token IDs. For this Burp Suite training tutorial, let us start with sending a request that contains a session token.

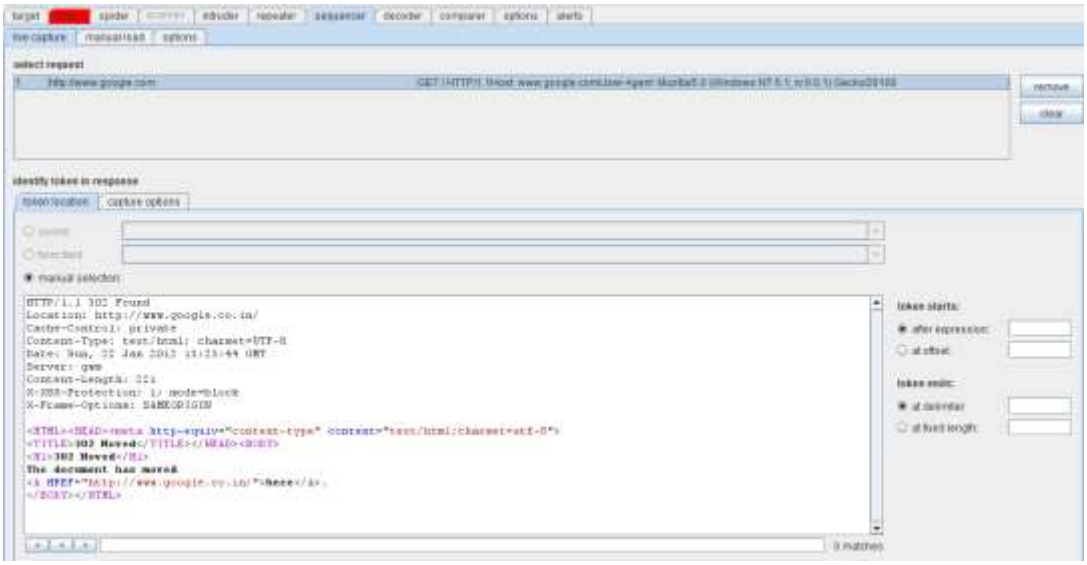

Figure 1. *Token request using sequencer (click to enlarge)*

Figure 1 shows a token request to the website google.com. The right side of the screenshot has the token start and token end expressions. You can either specify an expression such as "Google" or even set the offset from where the token has to start. This also applies to the token end panel, where you can set the delimiter, or specify a fixed length for the capture to start. After fixing these parameters, click START CAPTURE.

Figure 2. *Start capture action panel (click to enlarge)*

The *start capture* action panel is depicted in Figure 2. It sends requests to the target and gives detailed analysis of the randomness in the cookie tokens. You can pause or stop the analysis at any point. For this Burp Suite training tutorial, stop the scan midway and check out the results. The screenshot in Figure 3 explains the results better.

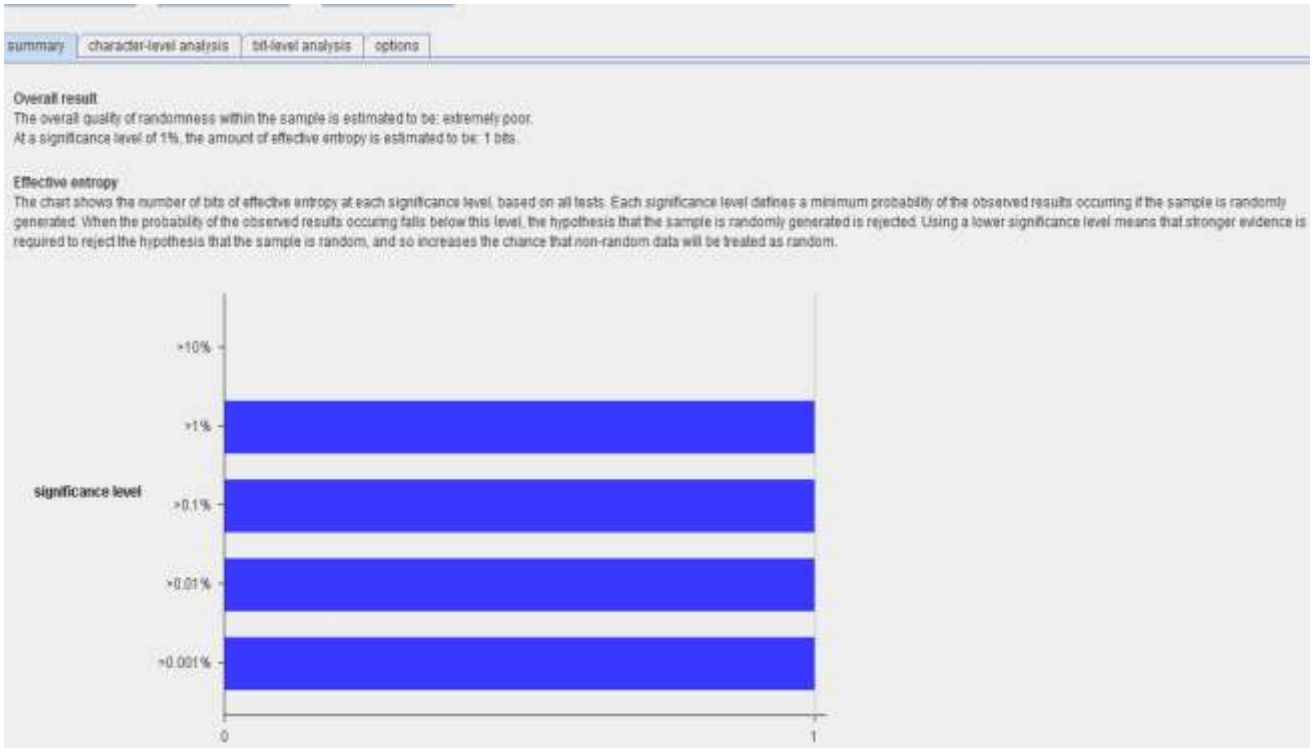

Figure 3. *Token randomness analysis results (click to enlarge)*

The scan components are as follows:

a. Overall result
b. Effective entropy
c. Reliability
d. Sample size

Burp automatically analyzes these aspects and generates this report in the sequencer tool. Burp also provides character-level analysis, which reports on the degree of confidence in the randomness of the sample, through a graphical display. Similarly, bit-level analysis can be performed at the bit level. There is an option to pad characters and also to decode in base64 if needed.

For this Burp Suite training tutorial, let us look at the following options provided by Burp sequencer. None of these is compulsory for analysis and they can be chosen or dropped as desired.

1. **Character count analysis :** This test analyzes the distribution of characters used within each token.

2. **Character transition analysis:** This test analyzes the transition of characters between successive tokens. Depending on the randomness of the characters, the transitional analytics vary.

FIPS monobit test

This test does an analysis of the positions of 0s and 1s at each bit position. If the generation is random, then the distribution is likely to be approximately equal.

1. **FIPS poker test:** This divides the bit sequence into consecutive and unique groups of four. The distribution is evaluated by a chi-square calculation method.

2. **FIPS runs test:** As the name suggests, the bit sequence is divided into runs of consecutive bits with the same value.

3. **FIPS long runs test:** Similar to FIPS runs test, this test analyzes the longest bit sequence with consecutive bits of the same value.

4. **Spectral tests:** This is an advanced method with complex statistical analytics. It treats a bit sequence as a point in multidimensional space and performs the analytics.

5. **Correlation test:** The tests described thus far analyze each bit in an isolated manner. The correlation test puts together these isolated results and presents the analytics by considering bits as a whole.

6. **Compression test:** This test works on the principle of the standard ZLIB compression technique. The bit sequences are compressed and the degree of compression is calculated. A higher degree of compression translates to a lower degree of randomness.

Burp decoder

The Burp decoder tool is used to send a request to the decoder. Within the decoder, there are multiple options to encode the request in various formats such as base64, URL, and so on. There are also options to convert it to hashes such as MD5 or SHA-1.

Figure 4. *Burp decoder screenshot (click to enlarge)*

Figure 4 depicts a Burp decoder request. For our Burp Suite training tutorial, consider an encoded request such as the one shown in Figure 5. The upper portion shows a request encoded in the base64 format while the lower one depicts the request decoded into plain text. While the entire request has been encoded here, you could also selectively choose a portion of the request to decode/encode.

Figure 5. *Encoded request (click to enlarge)*

This tool is useful when there is client-side encryption of username and password into commonly used hashes or encoders. The username or password field can be selectively decoded and the content then viewed in plaintext.

Burp comparer

Burp comparer is used for comparisons between two sets of data. For instance, the two sets could display responses to two different requests. The comparison can be performed either on a word scale (word by word) or bit by bit. Burp automates this process for the user and compares the two requests or responses accordingly. For this Burp Suite training tutorial, the comparison shown in Figure 6 is of two different requests to a website.

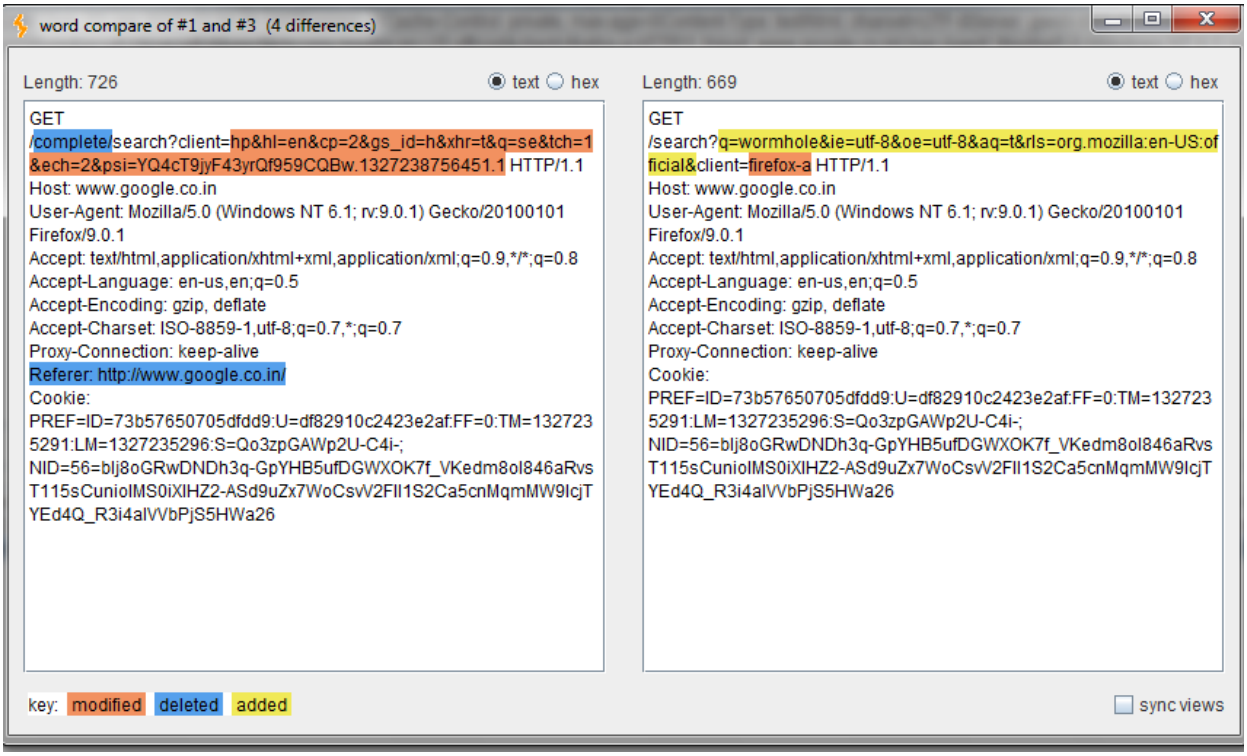

Figure 6. *Comparison of requests to a website (click to enlarge)*

This ends the Burp Suite training tutorial series. The extent to which Burp Suite can be used is limited only by the imagination of the user.

About the author: ***Karthik R*** *is a member of the NULL community. Karthik completed his training for EC-council CEH in December 2010, and is at present pursuing his final year of B.Tech. in Information Technology, from National Institute of Technology, Surathkal. Karthik can be contacted on rkarthik.poojary@gmail.com. He blogs at http://www.epsilonlambda.wordpress.com*

You can subscribe to our twitter feed at @SearchSecIN. Read the original story on SearchSecurity.in.