Parameter Pollution Vulnerabilities Detection Study Based on Tree Edit Distance

Yan Cao, Qiang Wei, and Qingxian Wang

National Digital Switching System Engineering & Technological R&D Center, Zhengzhou, China
vspyan@hotmail.com, funnywei@163.com, wqx2008@vip.sina.com

Abstract. A new web attack pattern called HTTP Parameter Pollution has been presented in recent years. The harm and detection method about HPP has become a hot topic in the field of web application security. In the paper, we started with analyzing the HPP attack pattern, researched on the necessary conditions and the potential harm of attack, pointed that the determination of parameter precedence is a prerequisite for the implementation and testing of such attacks, and proposed determination method for parameter priority based on tree edit distance to provide the necessary support for HPP vulnerabilities detection. As well as, we developed different detection methods for the difference of parameters between URL and the page. Finally the detection system for HPP vulnerability was realized, and some vulnerabilities have been discovered in real world.

Keywords: Web security, Parameter pollution, Parameter precedence, Vulnerability detection, Edit distance.

1 HTTP Parameter Pollution Attacks

HPP(HTTP Parameter Pollution attacks) was put forward first by Stefano Di Paola and Luca Carettoni in 2009[1]. Many well-known sites, such as Google, Yahoo, are found to have this vulnerability. HPP itself belongs to vulnerability of input validation. HPP along with XSS (Cross Site Scripting)[2], parameter tampering[3], SQL injection[4], in essence, dues to the lack of input validation. Through exploiting HPP vulnerability, attackers can modify the HTTP hard-coded parameters, change the behavior of Web applications, access or use the uncontrollable variables, as well as bypass input validation checks and WAF (Web Application Firewall Web Application Firewall) rules.

1.1 Priority of Parameter Selection

We focus on the type of HPP attack in the paper, which transfering the input information using URL results in. According to the syntax of the URL, "?" is the end of the

T.W. Chim and T.H. Yuen (Eds.): ICICS 2012, LNCS 7618, pp. 392–399, 2012.

file address to access, followed by the parameters passed to the server. Different parameters separated by "&". The variable name is before "=", and the parameter values is after "=". As follows:

?Username=jybox&pwd=password
?Do=&id=12

In order to avoid ambiguous when passed parameter contains special characters, a special character needs encoded in hexadecimal, as *%hh*. For example, if the parameters passed contain "?", and identifier "?" isolate the file addresses and parameters, "?" must be encoded for transmission, expressed as *%3F*.

When some parameters with identical name but different values, such as *example.pl?id=1&id=2&id=3*, the parameter *id* passing for three different values, it is need to study pre-conditions of HPP attacks that how the application deal with these. That is parameter priority.

Generally, there are three different ways which web application to deal with such issues in:

- Extract the first value as a parameter value. As *id = 1*.
- Extract the last value as the value of the parameter. As *id = 3*.
- Extract all values passed to the parameter as a list. For example, when the HTML page uses the Checkbox object to represent the form of a check box, several parameters with identical name but different values need be passed. To support this function, list of all the parameters as an array is passed in the majority of the programming language.

It is no unified standard to process the parameters with identical name but different values in different programming languages. Whatever method to choose, that is not HPP vulnerability. However, if the web application developer ignores such issues, it is likely for an attacker to exploit combining with a variety of ways. How the different parameter priorities impact HPP attack will be explained below.

1.2 HPP Attack

HPP can be described that given the existing legal parameter *p* and malicious parameters *p'*, if the *p'* together with the URL encoding *hh%* of the parameter separator are injected together *p*, becoming the new input parameters *p%hhp'*, and the application doesn't check legality of the parameter *p* and filter, *p'* would be accepted by the application and the attacker's intent would be achieved.

The following is to take an example for describing the actual process of HPP attack.

Scenario: there is one election web site URL: http://host/election.jsp?poll_id=4568, containing two candidates called White and Green. The URL of the election passes a single parameter *poll_id*. The server might use the following code:

ID = Request.getParameter ("pool_id")
href_link = "vote.jsp? poll_id =" + ID + "& candidate = xyz"
Two new links *Link1* and *Link2*:
Link1: <a href="vote.jsp?poll_id=4568&candidate=white"> Vote for Mr.White </ a>
Link2: <a href="vote.jsp?poll_id=4568&candidate=green"> Vote for Mrs.Green </ a>

Through these two links, the users can vote for two candidates.

When processing parameter *pool_id* did, server doesn't check necessarily, so *pool_id* can become an exploitable parameter. An attacker creates a new parameter injection URL, as follows:

http://host/election.jsp? poll_id = **4568% 26candidate% 3Dgreen**

At this point, decoded *pool_id* is *4568&candidate=green*. Thus two election links are generated, as follows:

Link 1': <a href = vote.jsp? Pool_id = 4568 **& candidate** = **green & candidate** = the white> Vote for Mr. White </ a>

Link 2': <a href = vote.jsp? pool_id = 4568 **& candidate** = **green & candidate** = green> Vote for Mrs. Green </ a>

Request.getParameter (candidate) in JSP only takes the first value when encounter some parameters with identical name but different values (determined by the priority of parameters). Therefore, whichever link the user clicks on, Green would be eventually elected. This is a complete instance of HPP attacks.

In this paper, we begined with analysis of HPP attacks, studied on attack conditions and testing methods, and pointed out the important influence of the parameters priority of HPP attack, and proposed the method for determining priority of the parameters based on the HTML tree edit distance. As well as more sound and viable solution about the HPP vulnerability detection was given.

2 System Architecture

According to the characteristics of HPP attacks, HPP vulnerability detection system based on tree edit distance was designed, consisting of five modules, namely web crawling module, HTML parsing module, parameters priority determination module, vulnerability scanning module and page response comparison module, as shown in Fig. 1.

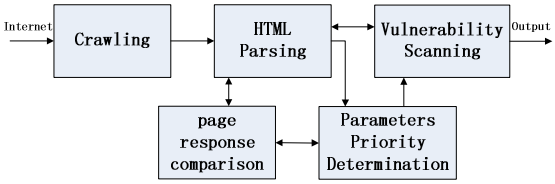

Fig. 1. HPP vulnerability detection system

The main function of each module is represented as follows:

- **Web crawling module** traversals web sites following the HTTP protocol, automatically extracts the hierarchy of the web sites, adds a URL link to the web's hierarchy to find these pages containing the passed parameters which can be browsed and interacted. From efficiency considerations, crawling depth is 3-layer.

- **HTML parsing module** obtains web pages from the web crawling module, of which DOM structure is parsed, and extracts all the links in the page and text and the URL of the form.
- **Parameters priority determination module** determines the response behavior of pages which receive some parameters with identical name but different values, in order to determine the parameter priority (see 3.1).
- **Vulnerability Scanning module** according to the parameter priority passed, injectes URL-encoded test parameters into the existing parameters in the page in the query string, checks whether there exist injected test parameters in link element of the response page, and action and hidden domain of form element, in order to detect whether there is the HPP vulnerability (see 3.2).
- **Page response comparison module** by HTML tree edit distance algorithm, determines whether the two pages are equivalent, to provide the basis for the parameters priority determination (see 3.1).

3 Key Algorithms and Technology

From Section 1, it can be drawn that the parameters priority determination is a prerequisite for HPP vulnerability detection. Parameters priority determination based on tree edit distance is the main contributions. Parameters priority determination and HPP vulnerability detection are represented in this section.

3.1 Parameters Priority Determination Based on Tree Edit Distance

Different web applications pass parameters with identical name but different values in different ways. This impact on the HPP attack has been discussed in Section 1. This section describes the problem to determine the parameters priority based on tree edit distance.

A. A Parameter Priority to Determine Algorithm

In this paper, the parameters priority determination method requires that one parameter need be transferred for three times. For example, the value *orgin* at the 1st get the server response page *P1*, 2nd with value *new* get page *P2*, and 3rd with value *orgin* and *new* to get page P3. As follows:

P1: pptest.php?par1=origin
P2: pptest.php?par1=new
P3: pptest.php?par1=origin&par1=new

By comparison among the three response pages, the parameters priority is determinated. The specific algorithm is shown in Fig. 2.

Parameter priority determination algorithm is explained as follows:

1: If the page *P1* and *P2* are equal, indicating that the identical parameter was transferred with both different values , and web application makes the same response, at this point, it is unable to determine the parameters priority.

2: If the page *P2* and *P3* are equal, indicating that web applications accepts the last value, selection strategy on some parameters with identical name but different values is to take the last value.

3: If the page *P1* and *P3* are equal, indicating that web applications accepts the first value, selection strategy on some parameters with identical name but different values is to take the first value.

```
Input: response page P1,P2,P3
Output: parameter priority or NULL
1：If(P1=P2) return NULL;
2：else if(P2=P3) return Last;
3：else if(P1=P3) return First;
4：else if（join匹配）return Join;
5：else {eliminate page;jump 2}
```

Fig. 2. Parameter priority determination algorithm

4: If the page *P1*, *P2*, *P3* are pair wise unequal, it is suspect that web application to accept all of the parameter values. Then the two values are connected into new string by a comma or space character string. And the new string is compared with page P3 strings. If be equal, it shows that web application can accept all of the parameter values.

5: If the page *P1*, *P2*, *P3* are pair wise unequal and web application does not accept all of the parameter values, it is possible that the page containing dynamic content result in the response page being not equal. At this point, the dynamic information contained in the response page should be disposed of first. Then go back to Step 2 of the algorithm.

Page equivalen means that the application made the same response to various requests. So, this paper realized comparison between pages based on tree edit distance algorithm.

B. Comparison Algorithm between Pages Based on Tree Edit Distance

HTML tree edit distance is used to quantify the similarity of the structure bewtween both pages. Edit distance, also known as Levenshtein distance, usually is used for similarity calculation between two strings[5].

Through parsing hierarchical tag structure, any HTML can be converted to an HTML DOM tree. Each node is identified corresponding to its HTML tag name. Each data object in HTML document can also be converted into a tree. Therefore, web page can be extracted into into HTML tree, and degree of similarity can be calculated between both pages by the tree edit distance.

HTML tree edit distance algorithm proposed in this paper improved on the STEDM algorithm[6]. Because the tree edit distance operations such as insertion, remove, and replacement, are on leaf node, the performance is enhanced.

Definition 1. HTML Tree Node Mapping: given tree *T1* containing *n1* nodes and tree *T2* containing *n2* nodes, mapping set *M (i, j)* between both trees satisfies *(i1, j1 is), (i2, j2)* with the following conditions:

(1) *i1 =i2* if and only if *j1=j2*.
(2) If *T1[i1]* is on the left of the *T1[i2], the T2[j1] is* on the *left* of *T2[j2]*.
(3) If the *T1[i1]* is the ancestor of *T1[i2], T2[j1]* is the ancestor of *T2[j2]*.

Definition 2. The Tree Edit Distance: If the tree *T1* is converted to tree *T2,* the tree edit distance *Dis(T1, T2)* is the number of operations for two trees to convert. As follows:

$$Dis(T1, T2)=Rep\times |R| +DEL\times|D|+Ins\times|I|$$

Among them, *Rep, Del, Ins* respectively represent operations of replace, delete, insert. HTML tree edit distance algorithm shown in Fig. 3:

```
Input: tree T1,T2
Output: edit distance dist
M is the number of T1's childnodes
N is the number of T2's childnodes
C1i is set of t1[i] deriving nodes
C2j is set of t2[j] deriving Nodes
Dis(T1, T2)
{
   for i = 1 to m
   for j = 1 to n
   {
    if (t1[i]==t2[j]) dist=0;
    else
    {
      if (t1[i] is leaf node)
      dist= dist+R*1+I*|C2j|;
      else if (t2[j] is leaf node);
      dist= dist+ R*1+D*|C1i|;
      else if(t1[i] and t2[j] aren't leaf
      node)
      dist=dist+Dis(t1[i], t2[j]);
     }
   }
   return dist;
}
```

Fig. 3. The HTML tree edit distance algorithm

By *Dist = the dist * (m + n) / 2 * mn* normalized, the range of the tree edit distance is [0,1]. From manual testing, if the edit distance of web pages is less than 0.15, they can be considered equal. If the edit distance is more than 0.8, they can be considered unequal. Ranging from 0.15 to 0.8, similarity between them needs to be further analyzed manually.

3.2 HPP Vulnerability Scanning

In this paper, the HPP vulnerabilities of the web application are detected, through injecting the URL-encoded test parameters into known legal parameters and detecting

behavior of the response page. If in link element of the response page, and action and hidden domain of form element, the injected test parameters are found out, it is considered that the page contains HPP vulnerability. The detection process is shown in Fig. 4.

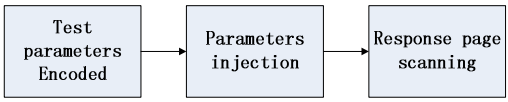

Fig. 4. HPP vulnerability scanning process

For example, the test parameter is encoded into *%26testparameter%3Dtp*, which is injected into the existing legal parameters *parameter=lp*. Then elements and domain on the answer page are scanned to serach for *&testparameter=tp*. If the test parameter exists and the parameter *lparameter* can be contaminated, the measured page contains HPP vulnerability.

Parameters in the pages and URL are not identical, so three situations should be discussed as follows:

- If parameters exist in the URL and the page, they can be injected directly to test.
- If parameters exist in the URL but not page, page may rename the parameters in the URL. So it is feasible to search the corresponding renamed parameters in the page and re-injection test.
- If parameters exist in the page and but not URL, the parameters should be explicitly place in URL and used for re-injection test.

4 Experiment

According to technical method in the paper, the HPP vulnerability detection system was designed to experimental test.

In order to verify the validity of the system on the HPP vulnerability detection, the system scanned for shopping, education, government, and search engine sites and found HPP vulnerabilities in the real Internet environment.

Take an example of HPP vulnerability on a search engine site. Fig. 5 is the normal search results page after entering the keyword *result*. Fig. 6 shows the page after the parameter *area* injected the test string into *area=0%to26kw%3Dtest*. Then the pages are the same as one before injected. However, when select Page 2 of the search results, as shown in Fig. 7, this should show search result with the key words *result*. But the page shown search result with a test parameter *test* instead. It is shown that through injecting the parameter *area* into the test string, due to selecting the last value of the priority strategy, the server accepted the parameter *test*, and achieved the purpose of polluting parameter *area*, to detect HPP vulnerability. In testing, the parameter *page* in URL is the same effect of pollution as *area*, but the parameters *do* and *src* are not.

There is a certain false rate in the system. Because after the test parameters were injected, it was accepted as a new parameter, it did not reach the effect of polluting other parameters. For axample, the parameter *%26test%3Dtp* injected into *lparameter=lp*, becames a new parameter *lparameter=lp%26test%3Dtp*, instead of *lparameter=lp&test=tp*. New parameter assignment resulting in page error is mistaken for HPP vulnerability. The system need be improved in the future.

Fig. 5. The normal search page

Fig. 6. The page after the injection parameters

Fig. 7. The page after the pollution parameters

5 Conclusion

Nowadays, web application has been widely applied. Increasingly complex interactions and the dynamic characteristics strengthen the function and role of Web applications, but it also brings many new security issues. Traditional SQL injection, XSS and other web attacks are still continuing today, and the new attack and web vulnerability mode will still continue to appear. Although HPP vulnerability exists soon, its harm should raise people's concern and attention.

We studied on determine method for the priority of parameters based on the HTML tree edit distance. From analysis on the characteristics of parameter in the URL and page, HPP vulnerability detection solution was proposed. Finally, HPP vulnerability detection system was designed and realized. The effectiveness of the parameters priority determination has been verified by the experiment and HPP vulnerabilities has been discovered in real world.

References

1. Carettoni, L., di Paola, S.: HTTP Parameter Pollution (EB/OL) (May 2009), `http://www.owasp.org/images/b/ba/AppsecEU09_CarettoniDiPaola_v0.8.pdf`
2. Rafail, J.: Cross-Site Scripting Vulnerabilities (EB/OL) (May 27, 2008), `http://www.cert.org/archive/pdf/cross_site_scripting.pdf`
3. Prithvi, B., Timothy, H., Nazari, S., et al.: NoTamper: Automatic Blackbox Detection of Parameter Tampering Opportunities in Web Applications. In: Computer and Communications Security, CCS (2010)
4. Chapela, V.: Advanced SQL Injection (EB/OL) (April 11, 2005), `http://www.owasp.org/images/7/74/Advanced_SQL_Injection.ppt`
5. Bille, P.: A survey on tree edit distance and related problems. Theoretical Computer Science 1(3), 217–239 (2005)
6. Yang, W.: Identifying Syntactic Differences Between Two Programs. Software-Practice and Experience 21(7), 739–755 (1991)