ENFORCING WEB APPLICATION AGAINST HTML BRUTE FORCE ATTACK

Dr. Samer Saeed Essa

AL_Rafidain University College

Computer Science Department

sammersaeed@yahoo.com

Abstract

Securing web applications is today's most common aspect of securing the enterprise. Web application hacking is on the rise with as many as 80% of cyber attacks done at web application level or via the web. Most corporations have secured their data at the network level, but have overlooked the crucial step of checking whether their web applications are vulnerable to attack. This paper is devoted for presenting a model to protect web pages that acquire passwords and user names against HTML brute force. Along this paper the model will be presented as same as the methodologies of implementation. Algorithms of implementation also will be presented to emphasize the idea where all HTTP packets are trapped and relayed down the stack of the TCP/IP by inserting before HTTP (Port 80) and all HTTP traffic of information that is captured to check for the parameter passing in web and this proposed technique is called Enforcing layer.

Keywords: Web applications, Web Authentication, Web Attacking, Brute Force Attack, HTTP (Port 80).

ENFORCING WEB APPLICATION AGAINST HTML BRUTE FORCE ATTACK

Dr. Samer Saeed Essa

فَرْض تطبيقِ الويبِ ضدّ هجومِ HTML Brute Force

د.سامر سعيد عيسى

كلية الرافدين الجامعة

قسم علوم الحاسوب

الملخّص

أن تأمين تطبيقاتِ الويبِ وضمان المنتج المنشور في الويب هو المجال الأكثر شيوعاً اليومُ . حيث أن الهجمات على تطبيقات الويبِ في إرتفاع وبنسبة 80 % مِنْ هجمات الإنترنتِ الأخرى على مستوى تطبيقات الويبِ أو من خلال الويب نفسه ، حيث أمنت معظم الشركات بياناتِها على مستوى الشبكةَ ، ولكن قد أغفلت خطوة مهمه وهي التحقق في ما أذا كانت التطبيقات على الويبِ عرضة للهجومِ. هذا البحثُ مكرّسُ لعرض نموذج حِماية صفحاتِ الويب التي تتطلب كلمة السر وأسم المستخدم لتجنب المهاجمة من قبل HTML Brute Force و على طول هذا البحث سيعرض النموذج كمنهجياتِ تطبيقية وأن تطبيق هذه الخوارزمية سوف تعرض لتؤكد فكرةِ كُلّ حُزَم HTML سيتم َحْصرها ونقلها في Stack خاص بالبروتوكول TCP/IP من خلال أضافة معلومات قبل الدخول ألى Port 80 وذلك لغرض فحص تلك المعلومات قبل تمريرها الى هذا Port وأن هذا الأسلوب المفترض في هذا البحث يدعى Enforcing Layer .

الكلمات المفتاحية : تطبيقات الويب ، تخويل الويب ، قرصنة الويب ، المهاجمه بقوه (بعمق) ، بوابة 80 لبروتوكول http .

Introduction

A web threat is any threat that uses the internet to facilitate cybercrime Web threats use multiple types of malware and fraud, all of which utilize HTTP or HTTPS protocols, but may also employ other protocols and components, such as links in email or IM, or malware attachments or on servers that access the Web. They benefit cybercriminals by stealing information for subsequent sale and help absorb infected PCs into both nets [1]. Web threats pose a broad range of risks, including financial damages, identity theft, loss of confidential information/data, theft of network resources, damaged brand/personal reputation, and erosion of consumer confidence in e-commerce and online banking. Figure (1) shows the International Statistics for the threats on the web [2]. Web sites now has members that could reach hundreds or sometimes thousands over the entire world, with such sites the possibility of guessing the password for a random client could be increased due to the probability theory. Brute force attack is a technique that could be used to sweep over a certain set of elements to hit a specific collection of the set elements [3,4]. The target is a certain collection from that set that may composes a key for a certain problem (i.e. a password). Web applications considered this issue and tried to enhance the password against brute force attack by, first, increasing the length of the password and second, increasing the length of the set that the password is composed from.

ENFORCING WEB APPLICATION AGAINST HTML BRUTE FORCE ATTACK

Dr. Samer Saeed Essa

Brute force attack now a day's step into the world of intelligence and the mechanisms just moved to other horizon by trying to collect information about the target subjected to the attack and may be use some dictionary method or other methods to creep into the system [2]. Some systems can easily protect themselves against this attack by limiting the tries to a certain number of fails and then back list the source of tries. This is a sustainable method with network of fixed source like GSM networks, where the authentication is tied to the device itself and can't be originated from different sources, but with the Internet, the story is totally different [5,6]. The attacker can easily change the source of attack at each try and use a lot of different user names, as it has been announced before, some sites has members of thousands of clients [2].

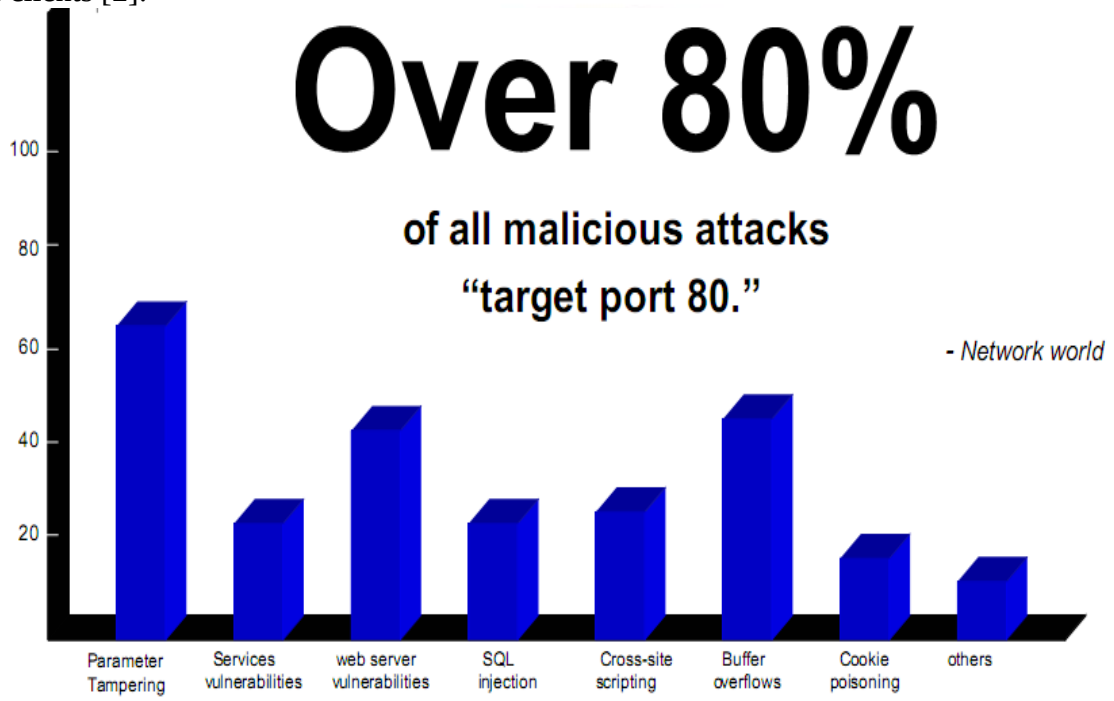


Figure (1): shows that international statistics for Web threats

1. Attacking Web Authentication

Authentication is a crucial element of any web application now days; the major authentication mechanisms in use on the Internet today are the user name and the password. How are such mechanisms attacked? In the next section, this paper will discuss techniques that can be used to exploit common vulnerabilities in Web authentication. The fact that authentication even exists for an

application suggests that the application developer has created some security infrastructure to prevent the casual hacker from easily obtaining access to other users' data. Hence, attacking Web authentication is not going to be a walk in the park. As always, however, it's the implementation that brings down the house [2,7].

1.1 Password Guessing

Password guessing is the most effective technique to Defeat Web authentication. Assuming there isn't some flaw in the selection of authentication protocol or its implementation, the most vulnerable aspect of most authentication systems is user password selection [7].

Password guessing attacks can be carried out manually or via automated means. Manual password guessing is tedious, but we find human intuition infrequently beats automated tools, especially when customized error pages are used in response to failed forms-based login attempts. in performing password guessing manually the attacker will sweep through some familiar pairs (Administrator/null, admin/admin … etc). With an automated tool, an entire dictionary of username/password guesses can be thrown at an application much more quickly than human hands can type them. Password guessing can be performed against almost all types of Web authentication.

1.2 Web Cracker

When encountering a page protected by Basic authentication in the consulting work, it will generally turn to Web Cracker to test account credential strength. Web Cracker is a simple tool that takes text lists of usernames and passwords (or combinations of both) and uses them as dictionaries to implement Basic auth password guessing. It keys on "HTTP 302 Object Moved" responses to indicate a successful guess, and it will find all successful guesses in a given username/password file (that is, it won't stop guessing once it finds the first valid account). Figure (2) shows the main page of Web Cracker. [2, 8]

ENFORCING WEB APPLICATION AGAINST HTML BRUTE FORCE ATTACK

Dr. Samer Saeed Essa

Figure (2): shows the main page of the Web Cracker application

1.3 Brutus

Brutus is a generic password guessing tool that comes with built-in routines for attacking HTTP Basic and Forms-based authentication, among other protocols like SMTP and POP3. Brutus can perform both dictionary attacks (based on pre-computed wordlists like dictionaries) and brute-force attacks where passwords are randomly generated from a given character set (say, lowercase alphanumeric). Figure (3) shows the main Brutus interface after performing a Basic auth password guessing attack.

ENFORCING WEB APPLICATION AGAINST HTML BRUTE FORCE ATTACK

Dr. Samer Saeed Essa

We are particularly impressed with the Forms-based auth attacker, primarily the Modify Sequence | Learn Form Settings feature. This allows you to simply specify a URL to a login form and Brutus automatically parses out the fields for username, password, and any other fields supported by the form (including hidden). Brutus also allows you to specify what responses you expect from the login form if a successful event occurs. This is important; because of the highly customizable nature of Forms auth, it is common for sites to implement unique response pages to successful or unsuccessful login. This is one of the primary impediments to successful password guessing against Forms-based auth. With the Brutus tool, you can customize password guessing to whatever responses the particular target site uses [2, 4].

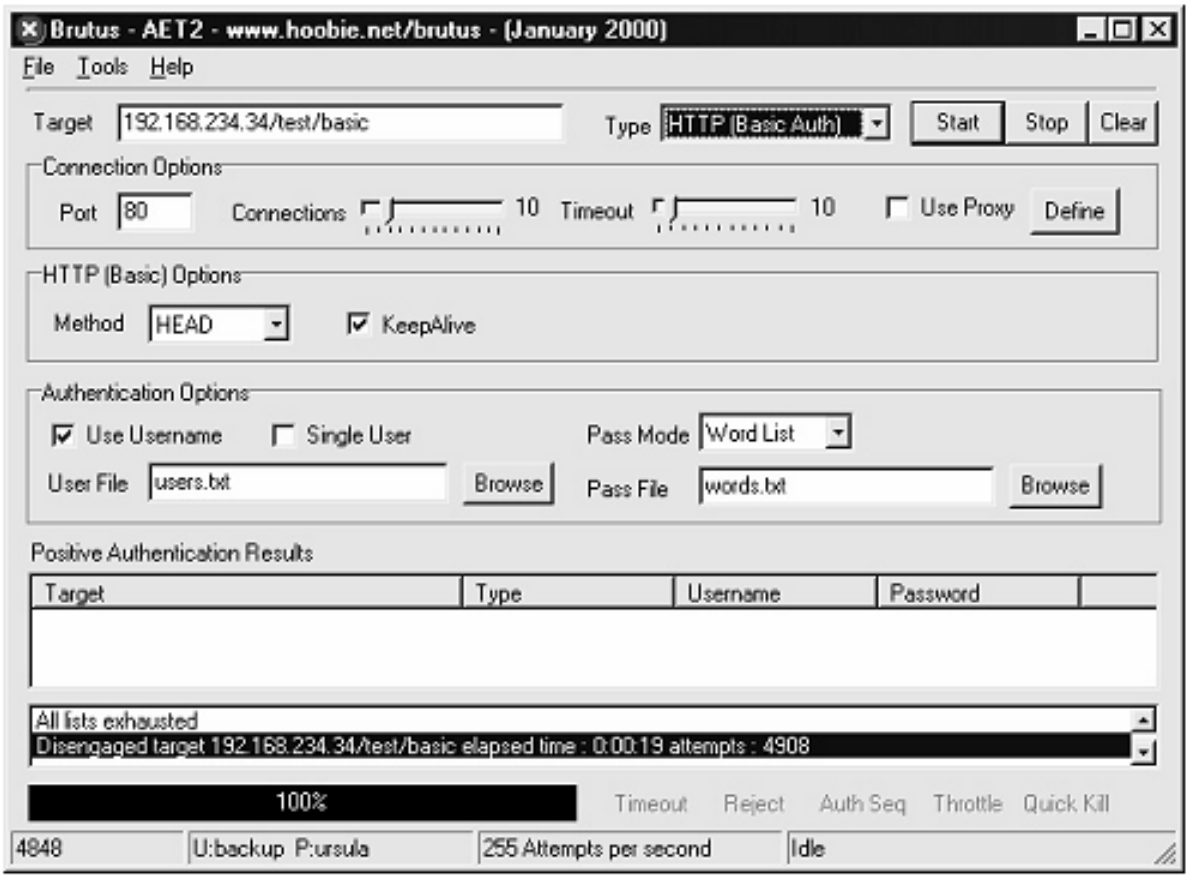


Figure (3): the main page of the Brutus application

The one thing that annoys us about Brutus is that it does not display guessed passwords when performing Forms auth attacks. We have also occasionally found that it issues false positive results, claiming to have guessed an account password when it actually had not. Overall, however, it's tough to beat the flexibility of Brutus when it comes to password guessing.

1.4 Session ID Prediction and Brute Forcing

Many e-commerce sites use a session identifier (session ID) in conjunction with Web authentication. A typical implementation stores a session ID once a user has successfully authenticated so that they do not need to retype credentials. Thus, if session identifiers are used in the authentication process, an alternative to attacking the passwords is to attack the session ID. Since the session ID can be used in lieu of a username and password combination, providing a valid session ID in a request would allow a hacker to perform session hijacking or replay attacks if the session ID is captured or guessed. The two techniques used to perform session hijacking are session ID prediction and brute forcing.

A secure session ID should be randomly generated to prevent prediction. However, many implementations do not follow this principle. Have seen many Web sites fall by using predictable, sometimes sequential, session identifiers. Many mathematical techniques such as statistical forecasting can be used to predict session identifiers.

The second technique for attacking session ID involves making thousands of simultaneous requests using all possible session IDs. The number of requests that need to be made depends on the key space of session ID. Thus, the probability of success of this type of attack can be calculated based on the size and key space of the session ID [2].

1.5 Countermeasures for Password Guessing

The most effective countermeasure against password guessing and brute forcing is a combination of a strong password policy and a strong account lockout policy. After a small number of unsuccessful login attempts, the application should lock the account to limit the exposure from this type of attack. However, be careful of denial-of-service attacks against an application with an excessively paranoid account lockout policy. A malicious attacker could try to lock out all of the accounts on the system. A good compromise that many application developers choose is to only temporarily lock out the account over the certain time [7, 8].

3. Brute Force Attack [9]

In cryptography, a brute-force attack, or exhaustive key search, is a strategy that can, in theory, be used against any encrypted data. Such an attack might be utilized when it is not possible to take advantage of other weaknesses in an encryption system (if any exist) that would make the task easier. It involves systematically checking all possible keys until the correct key is found. In the worst case, this would involve traversing the entire search space.

The key length used in the encryption determines the practical feasibility of performing a brute-force attack, with longer keys exponentially more difficult to crack than shorter ones. Brute-force attacks can be made less effective by obfuscating the data to be encoded, something that makes it more difficult for an attacker to recognize when he/she has cracked the code. One of the measures of the strength of an encryption system is how long it would theoretically take an attacker to mount a successful brute-force attack against it.

Brute-force attacks are an application of brute-force search, the general problem-solving technique of enumerating all candidates and checking each one.

4. The Proposed Hypothesis

The proposal presented in this paper is to prove a hypothesis that any rapid change in the incoming quires is a suspicious action and the changed part of the quires has a very high probability to be the password field especially if some replies from the server side are failed. The following figure (4) shows the basic idea of the hypothesis:

ENFORCING WEB APPLICATION AGAINST HTML BRUTE FORCE ATTACK

Dr. Samer Saeed Essa

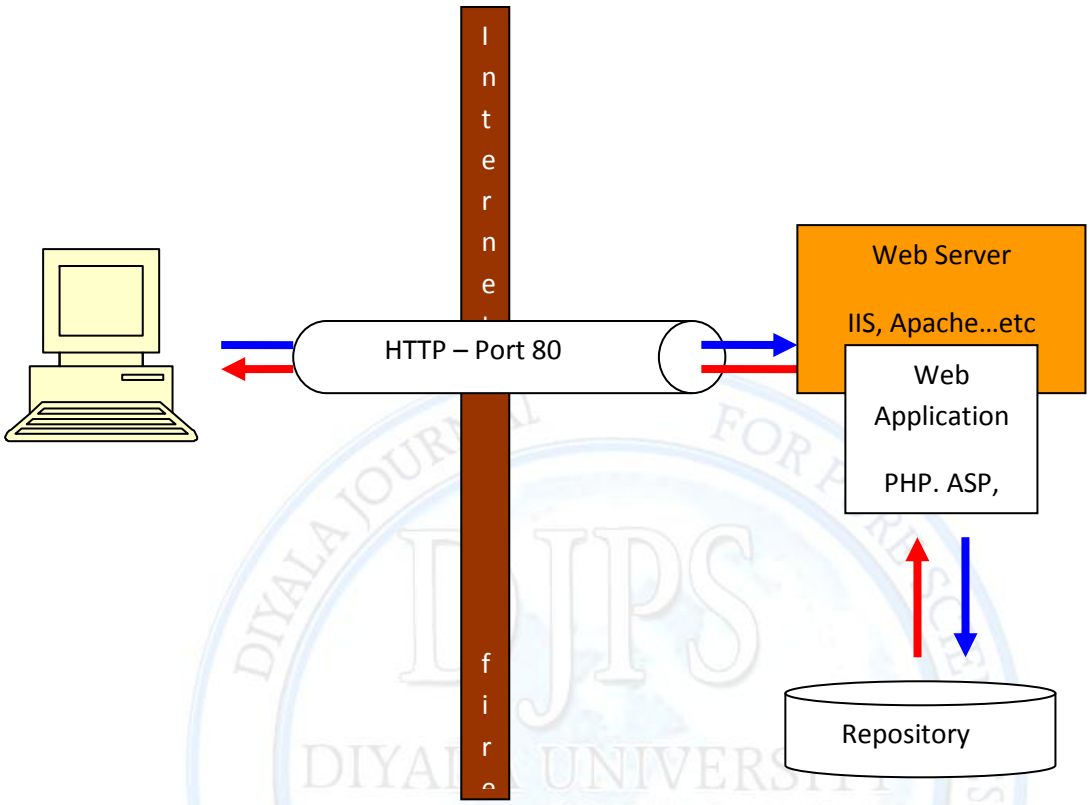


Figure (4): the basic idea of the hypothesis

The communication between the internet explorer and the web server is implemented through the HTTP and the policy is based on the request and response.

When the user requests a certain web page, the explorer will send an HTTP packet as the following:

GET /serverdirectory/filename

HTTP/1.1

Accept:*/*

Referrer: http://website

The above is for the static web page retrieval but the modern approach is by contacting active server pages (i.e., PHP, JSP, ASP…). These server pages have a program

that accepts the user request as an argument and response upon it with the corresponding action as an HTML text sent to the requester (client Internet Explorer).

A URL parameter is a name/value pair appended to a URL. The parameter begins with a question mark (?) and takes the form name=value. If more than one URL parameter exists, each parameter is separated by an ampersand (&). The following example shows a URL parameter with two name/value pairs:

http://server/path/document?name1=value1&name2=value2

In this paper inserted a filter driver before HTTP (Port 80) and all HTTP traffic are captured to check for the parameter passing that is called enforcing layer as show in figure (5)

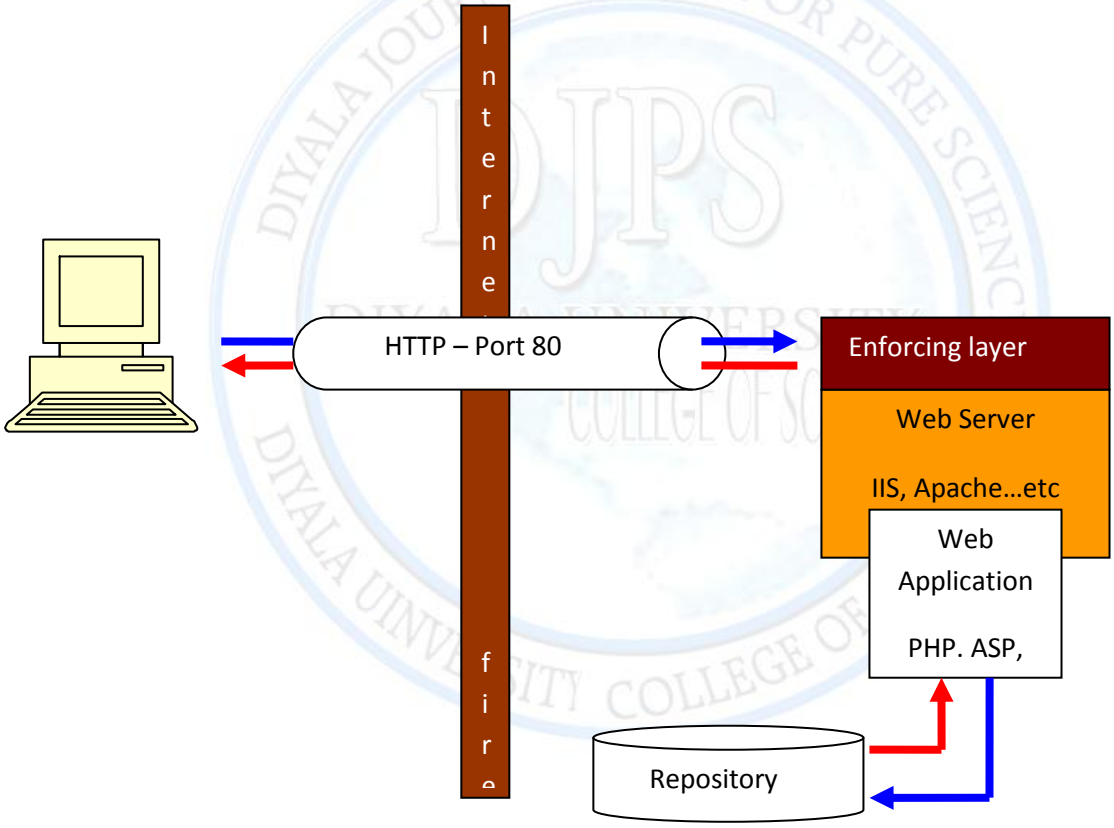


Figure (5): Insert the Enforcing Layer

4.1 The Proposal Algorithms

In this section present the two algorithms which one that are implement to check the suspicious brute force and another algorithm make to detect the brute force attack.

4.1.1 Suspicious Brute Algorithm

This algorithm implement to check the suspicious brute force by initializing buffer with 8 mega bytes with array called timer and capture the HTTP packet to check that is present bellow:

1- Initialize Buffer Area with 8 mega byte
2- Initialize timer array
3- Capture the next HTTP packet
4- If the requested resource is not an Active Server Page with arguments, goto 10
5- Register the resource name , increase the resource reference counter
6- Timer [resource reference counter] = GetSystemTime.
7- If resource reference counter < brute threshold, goto 3
8- Calculate the standard deviation for the differences in the arrivals of the requests for the resource
9- If the standard deviation is slow , <= 0, goto 11
10- Serialize all captured packet to HTTP (Port 80).
11- Start brute attack detected

4.1.2 Brute Attack Detected Algorithm

This algorithm implement of detected the brute force attack by checking the request resource with the register resource that is present bellow:

1- If the requested resource within the registered resources, goto 5
2- Redirect the requested resource (i.e., Web page) to a fake resource
3- Register the resource that the brute attack detected
4- Restart the Suspicious brute
5- Send member confirmation form

5. The Application of Proposed System

In this section is presenting the information flow when clients interact web application; we mean here the web applications that are under the monitoring of the proposed system through the essential components, show that in figure (6).

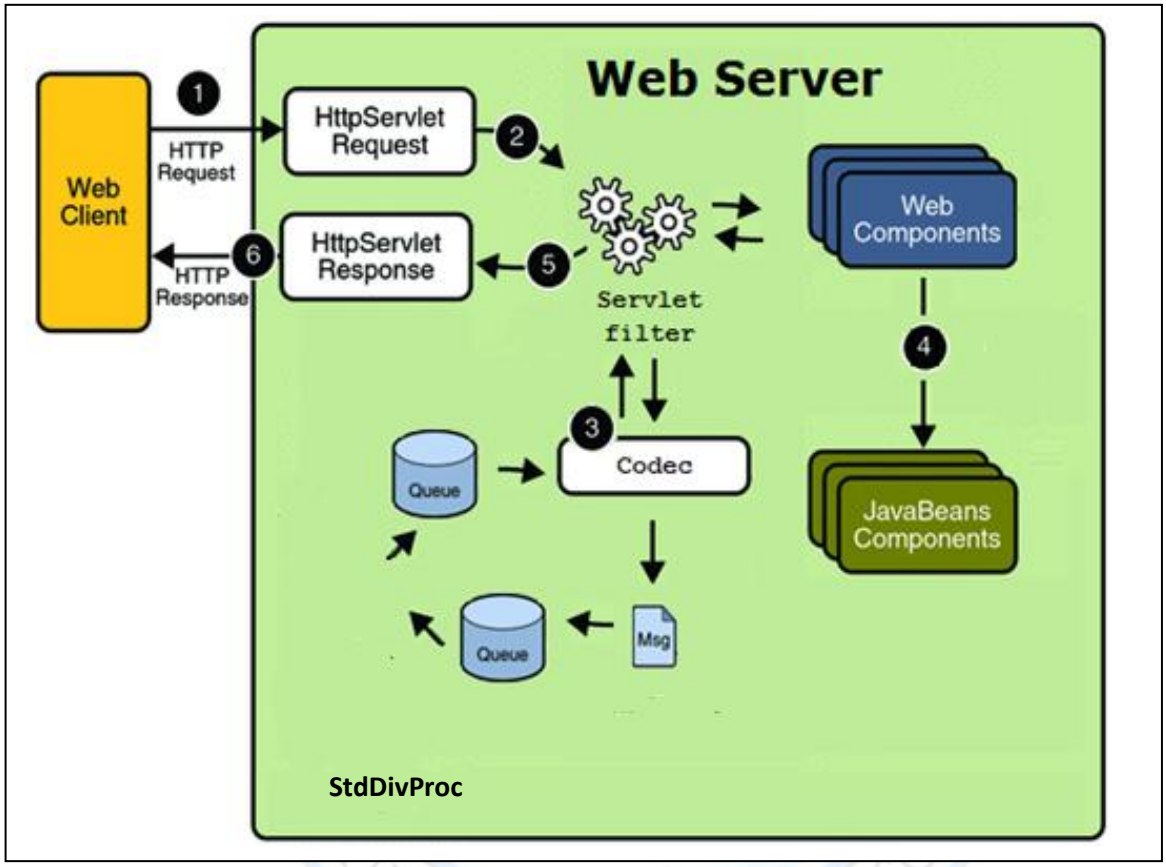


Figure (6) The essential components of proposed system

The essential components of proposed system are:

1- **Web client:** it is the internet explorer used to browse for the web application; any internet explorer can be used such as firefox, internet explorer, safari, …, or others.
2- **Web application server:** the platform used to host web application; web application has to cope with web application server in order to run smoothly, many web application servers are available such as Glassfish and TomCat. This thesis is using TomCat due to its light weight and performance despite that fact that TomCat does not support EJB by default.
3- **JSP pages:** these are the front edge of the web application and are to be accessed over the internet using the web client software. Each JSP page is assigned a unique URL (Unified Resource Locator) or URI (Unified Resource Identifier).
4- **Servlets:** these are the real business logic to process incoming client request and present responses to web client applications. Servlets are running within a servlet context which is the directory in which the servlet is running.
5- **Web filter:** it is an object installed to intercept client request before it gets to servlet and JSP; web filter has been used to forward incoming traffic to Agent behaviours
6- **HttpServletRequest:** this is an object in which clients information is encapsulated. Any data passed from web client to web server is packed into the HttpServletRequest object. It is the responsibility to web application programmer to intercept this object and extract passed parameters.
7- **HttpServletResponse:** this is an object in which information is passed back to the web client software.

Web filter or in other words the servlet filter is installed at the server side to capture all traffic between client side and server side. HTTP packets are interpreted and all parameters are revealed. User name and password are normally hidden input an element added to HTML page and then in the figure (7) presents the semantics used to interpret HTTP packet:

ENFORCING WEB APPLICATION AGAINST HTML BRUTE FORCE ATTACK

Dr. Samer Saeed Essa

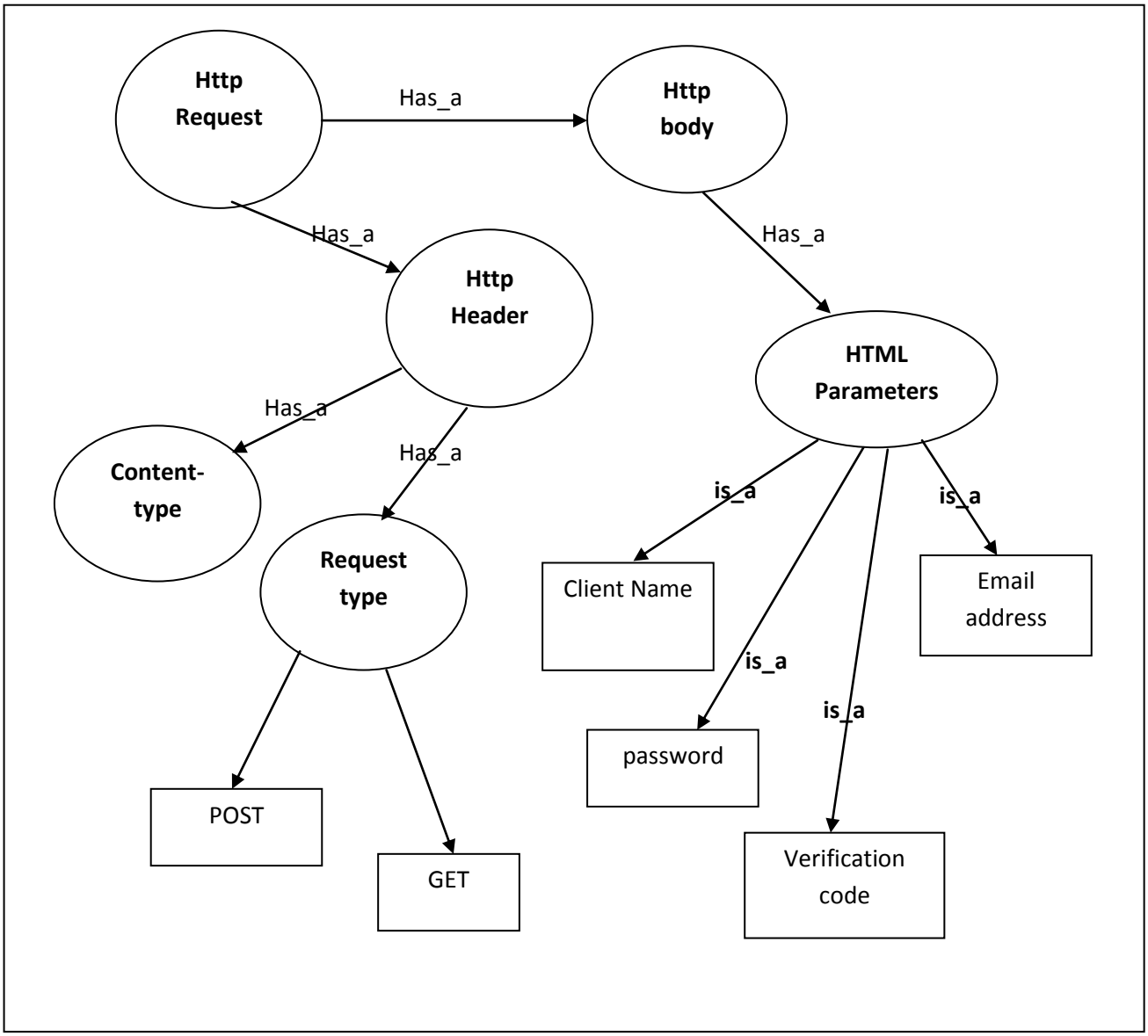
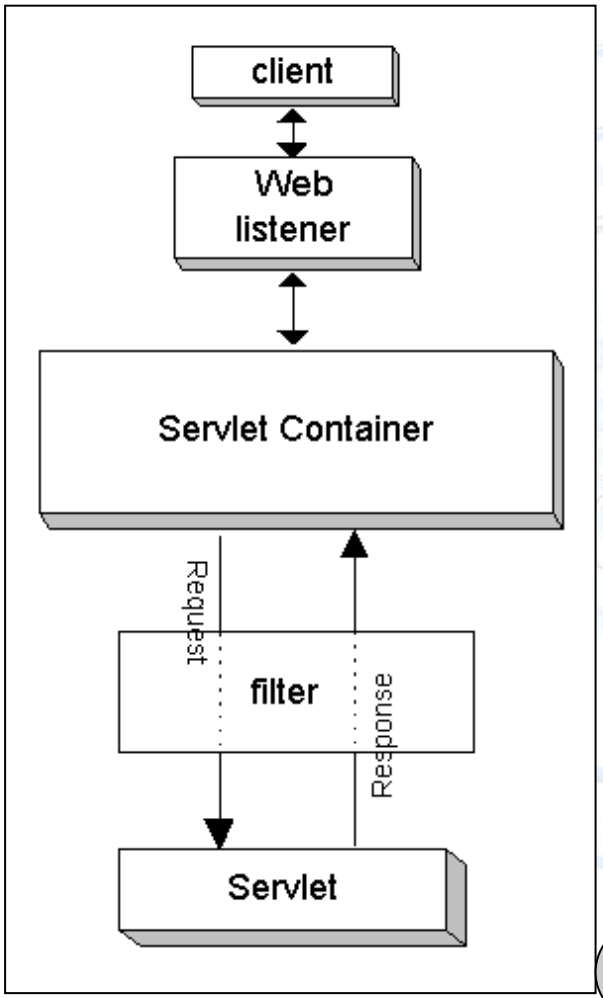


Figure (7): Ontology used to interpret HTTP request packet

ENFORCING WEB APPLICATION AGAINST HTML BRUTE FORCE ATTACK

Dr. Samer Saeed Essa

HTTP packets are captured by web filter by installing servlet filter as it is presented by figure (8), where this filter is responsible on retrieving parameters accompanied request coming from client browsers.




```
Public void dofilter(ServletRequest
request, ServletResponse response,
FilterChain chain)        {

  doBeforeProcessing(request, response);

   String name = null;

   If((name = request.getParameter())  !=
NULL     && password =
request.getParameter() != NULL)    {

        RequestDispatcher rDisp =
request.getRequestDispatcher("
StdDivProc");
rDisp.forward(request.response)
```

Figure (8): Servlet Filter installed to Capture HTTP packet and forward it to other Processing Elements

Conclusions

1- HTML based brute attack can be detected upon rapid time change for the requesting packets for a certain resources.
2- The Attacker can change the time interval but this will make the attack infeasible for him.
3- Server side programmers should consider long time brute force attack and change passwords corresponding matrices.
4- Brute force attack might hit a success with less tries, this is a big problem and there should be considered in different manner.
5- The site with huge number of members has the largest potential to lie down be brute force.
6- The standard deviation dose not effect the speed of the internet response if the following rule has been satisfies: Tprocess < Tinternet , where Tprocess: is the time needed to process standard deviation on local computer and Tinternet: is the time needed to transfer the reply to requester, in other words the speed of reply. (Broadly speaking, in the modern machine this rule is always satisfied).

References

[1] Andrew S. Tanenbaum, Vrije University, " Modern Operating Systems", Prentic-Hall, Inc. 2001

[2] Joel Scambray and Mike Shema, "Web Application Security Secrets & Solutions", McGraw-Hill, 2002

[3] P. J. Deitel and H. M. Deitel, "Internet & World Wide Web , How to Program", Pearson Education, Inc. 2009

[4] Shreeraj Shah, "Advanced Web Hacking", EUSecWest conference in London, 2006

[5] IRV Englander, Bentley College, "The Architecture of Computer Hardware And Systems Software", John Wiley and Sons. Inc., 2003

[6] P.J. Deitel and H.M. Deitel, " Java, How to Program", Pearson Education, Inc. 2005

[7] Mount Ararat Blossom, " Securing IIS by Breaking", preceding of (http://hackbbs.org), 2000

[8] P.J. Deitel and H.M. Deitel, " Java Web Services for Experienced Programmers",Pearson Education, Inc. 2003

[9] Wikimedia Foundation,"Brute Force Attack", www.wikimediafoundation.org , 30 March 2012.