



1

2

# Application Vulnerability Description Language v1.0

3

4

## OASIS Standard, May 2004

5

### Document identifier:

6

AVDL Specification - 01

7

### Location:

8

<http://TBD>

9

### Editor:

10

Jan Bialkowski, NetContinuum, [jan@netcontinuum.com](mailto:jan@netcontinuum.com)

11

Kevin Heineman, SPI Dynamics, [kheineman@spidynamics.com](mailto:kheineman@spidynamics.com)

12

### Contributors:

13

Carl Banzhof, Citadel

14

John Diaz, Lawrence Livermore National Laboratory

15

Johan Strandberg, NetContinuum

16

Srinivas Mantripragada, NetContinuum

17

Caleb Sima, SPI Dynamics

18

### Participants:

19

Jeremy Poteet, Individual

20

Lauren Davis, Johns Hopkins University Applied Physics Laboratory

21

Andrew Buttner, Mitre Corporation

22

Gerhard Eschelbeck, Qualys

23

Jared Karro, Bank of America

24

Montgomery-Recht Evan, Booz Allen Hamilton

25

Ajay Gummadi, Individual

26

Yen-Ming Chen, Individual

27

Brian Cohen, SPI Dynamics, Inc.

28

John Milciunas, SPI Dynamics, Inc.

29

Matthew Snyder, Bank of America

30

Chung-Ming Ou, Chunghwa Telecom Laboratories

31

Anton Chuvakin, Individual

32

Nasseam Elkarra, Individual

33

Roger Alexander, Individual

34

J. Wittbold, Mitre Corporation

35

Lluis Mora, Sentryware

36

### Abstract:

37

This specification describes a standard XML format that allows entities (such as

38

applications, organizations, or institutes) to communicate information regarding web

39 application vulnerabilities. . Simply said, Application Vulnerability Description Language  
40 (AVDL) is a security interoperability standard for creating a uniform method of describing  
41 application security vulnerabilities using XML.

42

43 With the growing adoption of web-based technologies, applications have become far  
44 more dynamic, with changes taking place daily or even hourly. Consequently, enterprises  
45 must deal with a constant flood of new security patches from their application and  
46 infrastructure vendors. . To make matters worse, network-level security products do little  
47 to protect against vulnerabilities at the application level. To address this problem,  
48 enterprises today have deployed a host of best-of-breed security products to discover  
49 application vulnerabilities, block application-layer attacks, repair vulnerable web sites,  
50 distribute patches, and manage security events. Enterprises have come to view  
51 application security as a continuous lifecycle. Unfortunately, there is currently no  
52 standard way for the products these enterprises have implemented to communicate with  
53 each other, making the overall security management process far too manual, time-  
54 consuming, and error prone.

55

56 Enterprise customers are asking companies to provide products that interoperate. A consistent  
57 definition of application security vulnerabilities is a significant step towards that goal. AVDL fulfills  
58 this goal by providing an XML-based vulnerability assessment output that will be used to improve  
59 the effectiveness of attack prevention, event correlation, and remediation technologies.

60

61 **Status:**

62 This document is an OASIS standard. Please send comments to the editors.

63

64 Committee members should send comments on this specification to [avdl@lists.oasis-](mailto:avdl@lists.oasis-open.org)  
65 [open.org](mailto:avdl@lists.oasis-open.org). Others should subscribe to and send comments to [avdl-comment@lists.oasis-](mailto:avdl-comment@lists.oasis-open.org)  
66 [open.org](mailto:avdl-comment@lists.oasis-open.org). To subscribe, send an email message to [avdl-comment-request@lists.oasis-](mailto:avdl-comment-request@lists.oasis-open.org)  
67 [open.org](mailto:avdl-comment-request@lists.oasis-open.org) with the word "subscribe" as the body of the message.

68

69 For information on whether any patents have been disclosed that may be essential to  
70 implementing this specification, and any offers of patent licensing terms, please refer to  
71 the Intellectual Property Rights section of the AVDL Technical Committee (AVDL TC)  
72 web page (<http://www.oasis-open.org/committees/avdl/ipr.php>).

73

74 **Eratta:**

75 The errata page for this specification is at: [http://www.oasis-](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=avdl)  
76 [open.org/committees/tc\\_home.php?wg\\_abbrev=avdl](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=avdl).

77

78

---

## Table of Contents

79	Introduction .....	4
80	1.1 Notations and Terminology .....	5
81	1.1.1 Notations .....	5
82	1.1.2 Terminology .....	5
83	1.2 Requirements .....	6
84	1.3 Out of Scope .....	6
85	2 AVDL Output .....	8
86	2.1 AVDL File Root .....	8
87	2.2 Traversal .....	8
88	2.2.1 Traversal Container .....	9
89	2.3 Vulnerability Probe .....	10
90	2.3.1 Vulnerability Probe Container .....	11
91	2.3.2 Vulnerability Properties .....	13
92	2.3.3 Vulnerability Specific .....	14
93	Appendix A. Acknowledgments .....	16
94	Appendix B. Revision History .....	17
95	Appendix C. Notices .....	18
96		

---

## 97 Introduction

98 The goal of AVDL is to create a uniform format for describing application security vulnerabilities.  
99 The OASIS AVDL Technical Committee was formed to create an XML definition for exchanging  
100 information about the security vulnerabilities of applications exposed to networks. For example,  
101 the owners of an application use an assessment tool to determine if their application is vulnerable  
102 to various types of malicious attacks. The assessment tool records and catalogues detected  
103 vulnerabilities in an XML file in AVDL format. An application security gateway then uses the AVDL  
104 information to recommend the optimal attack prevention policy for the protected application. In  
105 addition, a remediation product uses the same AVDL file to suggest the best course of action for  
106 correcting the security issues. Finally a reporting tool uses the AVDL file to correlate event logs  
107 with areas of known vulnerability.

108

109 In order to define the initial standard, the AVDL Technical Committee focused on creating a  
110 standard schema specification that enables easy communication concerning security  
111 vulnerabilities between any of the various security entities that address Hypertext Transfer  
112 Protocol (HTTP 1.0 and HTTP 1.1) application-level protocol security. Future versions of the  
113 standard will continue to add functionality until the full vision of AVDL is achieved. AVDL will  
114 describe attacks and vulnerabilities that use HTTP as a generic protocol for communication  
115 between clients and proxies/gateways to other Internet systems and hosts. Security entities that  
116 might use AVDL include (but are not limited to) vulnerability assessment tools, application  
117 security gateways, reporting tools, correlation systems, and remediation tools. AVDL is not  
118 intended to communicate network-layer vulnerability information such as network topology, TCP  
119 related attacks, or other network-layer issues. Nor is AVDL intended to carry any information  
120 about authentication or access control; these issues are covered by SAML and XACML.

121

122 Applications that use HTTP and HTML as their foundation access and communication scheme  
123 are vulnerable to various types of malicious attacks. The goal of the AVDL is to define a language  
124 for conveying information that can be used to protect such an application. This information may  
125 include (but is not limited to) vulnerability information as well as known legitimate usage  
126 information.

127

128 Vulnerability information may include:

- 129 • Discrete, previously known vulnerabilities against the application's software stack or any  
130 of its components such as operating system type/version, application server type, web  
131 server type, database type, etc.
- 132 • Information on an application's known legitimate usage schemes such as directory  
133 structures, HTML structures, legal entry points, legal interaction parameters, etc.

134

135 AVDL is capable of describing either type of information.

136

## 137 1.1 Notations and Terminology

### 138 1.1.1 Notations

139 The Keywords “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,”  
140 “SHOULD NOT,” “RECOMMENDED,” “MAY,” “MAY NOT,” and “OPTIONAL” in this document are  
141 to be interpreted as described in RFC 2119.

142

### 143 1.1.2 Terminology

- 144 • **AVDL** – This is an acronym for Application Vulnerability Definition Language. This is the  
145 abbreviated name for the standard XML format to be used by entities (e.g., applications,  
146 organizations, or institutes) to communicate information regarding web application  
147 vulnerabilities. Simply said, AVDL is a security interoperability standard, the goal of which is  
148 to create a uniform way of describing application security vulnerabilities using XML.
- 149 • **AVDL Version** – This field identifies the version number of the schema that is being used. As  
150 the AVDL standard evolves, each release of the standard will contain a unique version  
151 number.
- 152 • **Classification** – This identifier is contained within the vulnerability description. It identifies  
153 metadata regarding the vulnerability. Data such as the classification name and the severity  
154 value are part of the classification.
- 155 • **Description** – This descriptor contains a detailed description of the vulnerability. It will be  
156 used in report output to the user.
- 157 • **Expect Status Code** – This is the expected result from the server that was attacked. If the  
158 server response is different from the expected response, a vulnerability is identified.
- 159 • **HTTP Transaction** – Contains the request and response that the Test Script made.
- 160 • **Recommendation** – This descriptor contains information related to actions that could be  
161 taken to remediate the vulnerability. This may include patch information or other information  
162 related to the recommendation.
- 163 • **Remedy Description** – This is a container of the patch description. It may also include  
164 specific instructions to load the patch.
- 165 • **Remedy vulnID** – This identifier describes the specific remedy that will be required to resolve  
166 the vulnerability.
- 167 • **Session ID** – This is the identifier of the specific attack session. A session will contain one to  
168 many Traversal Steps (see Traversal Step ID). Each Session will be identified with a unique  
169 identifier. The session will contain a target and a date-time stamp for when the session  
170 begins.
- 171 • **Summary** – This descriptor defines a short summary of the vulnerability within the Test  
172 Probe.
- 173 • **Test Description** – This descriptor contains the attack that was used to identify the  
174 vulnerability.
- 175 • **Test Probe** – This is a container of the session that identified the vulnerability. The Probe  
176 contains both the raw request and raw response as well as parsed request and parsed  
177 response.
- 178 • **Test Script ID** – This descriptor identifies the test that was conducted as part of the Test  
179 Probe to identify the vulnerability. A Test Probe may contain one to many Test Scripts.

- 180
- 181
- 182
- **Traversal Ste** – A traversal is the sum of a request to a web server and a response from the web server. Each Traversal Step is identified with a unique identifier. The Traversal Step contains both the raw and parsed content of the request and response.
- 183
- **Vulnerability Description Title** – This descriptor defines the vulnerability within the Test Probe.
- 184
- **Vulnerability Probe** – This is a container for the Test Probes and may contain one to many Test Probes. The term “Probe” is used since the application originating the data is generic (e.g., assessment, protection, remediation, event correlation).
- 185
- 186
- 187
- 188

## 189 1.2 Requirements

190 The Application Vulnerability Description Language uses XML to support communication between  
191 applications that exchange information about web application vulnerabilities. Specifically the  
192 specification includes two major sections: Traversal and Vulnerability Probe.

193

194 The Traversal is a mapping of the structure of the site. Its purpose is to fully enumerate the web  
195 application. The Traversal is populated by assessment products to map the application and  
196 create a baseline of the site. It describes the requests and responses that were made to the  
197 server and the pages that were displayed as a result of the requests.

198

199 The Vulnerability Probe is a description of a vulnerability. It includes information about the  
200 vulnerability as well as how the vulnerability was found and, when possible, how it can be fixed.

201

## 202 1.3 Out of Scope

203 AVDL has been developed to describe web application vulnerabilities. It is not intended to be  
204 used to describe other types of vulnerabilities. This includes (but is not limited to) server,  
205 operating system, TCP related attacks, or other network layer issues. While vulnerabilities of  
206 these types may also fit within the AVDL model, the standard was not specifically developed for  
207 these types of vulnerabilities.

208

209 AVDL is not intended to carry any information about authentication or access control. These  
210 issues are covered by SAML and XACML.

211

212 Version 1.0 of the standard is specific to English language output. Future versions of the standard  
213 are anticipated to address or accommodate other languages.

214

215 Encapsulating well-defined behavior of the target application within the standard is not within the  
216 scope of AVDL version 1.0. Well-defined behavior is specific information relating to how the web  
217 application works. For example, valid values for a page as well as the behavior of the application  
218 with regards to invalid values. Discrepancies to this normal behavior would be identified as  
219 vulnerabilities. Future versions of the standard may address this issue.

220

221 A complete catalog of the potential vulnerabilities is not included in the specification. The  
222 standard will not contain any descriptors that contain any vulnerability storage containers. This  
223 includes either content or a list of identifiers (such as CVE).

224

225 This version of the AVDL standard addresses only web application vulnerabilities. Future versions  
226 of the standard may incorporate the output from other vulnerability scanners that are not web-  
227 based such as ISS and other probes.

---

## 228 2 AVDL Output

229 The purpose of this section is to articulate the output that AVDL generates using an example.  
230 This particular example is a "Translate: f" vulnerability. This vulnerability is a common web  
231 application vulnerability in IIS that allows remote attackers to view source of offered server-side  
232 scripts supported by IIS by using a malformed "Translate: f" header.

233 Throughout this section, the example XML is a sample of the Translate: f vulnerability output  
234 produced by AVDL. The complete example is contained in an appendix. In addition, where the  
235 Translate: f example does not apply, generic information was included in the example.

236

### 237 2.1 AVDL File Root

238 The beginning of the AVDL output contains a file root that includes information within the AVDL  
239 output. It is a metadata container to provide context for the rest of the file. The information  
240 contained in the file root includes the version of AVDL that is being used, the provider or vendor  
241 name that generated the output as well as URIs pointing to the OASIS standards body.

242

```
243 <avdl version="0.1-2003-09-27" provider="SPI"  
244 xmlns="urn:oasis:names:tc:avdl:0.0:mailto:avdl@oasis-open.org?:avdl:2003-09-27:a"  
245 xmlns:xhtml="http://www.w3.org/1999/xhtml"  
246 xmlns:avdl="urn:oasis:names:tc:avdl:0.0:names:mailto:avdl@oasis-open.org?:2003-09-27"  
247 xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

248

249 AVDL can be thought of in hierarchal terms. The highest level (or root) contains all the activity  
250 articulated through AVDL. The root container may contain multiple sessions. A session should be  
251 thought of as an action a user takes. For example, crawling a web site or scanning a web  
252 application for vulnerabilities are examples of sessions. Each session can contain one to many  
253 traversals. A traversal is a single request and response to and from a web server. Each traversal  
254 can be broken down into its raw and parsed form.

255

256 To keep this example simple, it contains only one session with one traversal and one  
257 vulnerability. The details of this example are explained in this section. Please refer to the AVDL  
258 schema for a complete description of the standard.

259

### 260 2.2 Traversal

261 The AVDL output is divided into two major sections. The first is the Traversal. This output reflects  
262 the basic structure of the site. It describes the requests and responses that were made to the  
263 server and the pages that were displayed as a result of the requests. A Traversal is a single  
264 transaction containing one or more request/response exchanges, each exchange is enclosed in a  
265 separate Traversal Container. These Traversal Containers provide a complete hierarchal  
266 description for a Traversal within a session.

267

268 The following is an example of a traversal session header. It contains the ID of the session with  
269 which it is associated, the target URI that was crawled, when the activity was started, and the

270 sequence number (a number designating this session in the ordered sequence of nodes visited  
271 during the crawl).). It also contains the raw request and response and the parsed request and  
272 response.

273

```
274 <session id="traversal-session" target="http://172.16.50.31" session-start="2004-02-  
275 10T16:57:25">  
276   <traversal-step time-stamp="2004-02-10T16:57:25" sequence-number="1"  
277   uri="http://172.16.50.31:80/">
```

278

279 It is important to note that the parsed header information contains query rules and content rules.  
280 Query rules define how the query is created. Content rules define what content will be filtered in  
281 the traversal. Since this example does not contain any content rules, all content will be displayed.

282

## 283 2.2.1 Traversal Container

284 The Traversal Container represents the request and the response for the round-trip HTTP  
285 traversal to the server. Each HTTP traversal is a request/response pair. While each Traversal  
286 Container contains only one request and response, a Session may contain many Traversal  
287 Containers. In general, to complete a single round trip, a traversal may encompass multiple  
288 protocols, each of which will contain its own request/response pair.

289

290 Within the standard, each request/response pair is represented in both raw and parsed form.  
291 Traversal Containers are listed in chronological order. In addition, each container can have its  
292 own specific rules. These rules are also captured within the Traversal Container.

293

294 The example shows the request and response completely in both the raw and parsed format.  
295 Content in this example contains h-refs, one of the children of the content container.

296

297 The request method includes the type of request, how the connection was made, what host was  
298 targeted, what URI was requested, and what protocol version was made. Following this  
299 information, the raw request is listed and then the parsed request. The request and response is  
300 parsed into header name and value pairs. In addition, the Query portion of the parsed information  
301 provides validation of the query. This validation could be applied for both the header and content.  
302 Like the parsed information, query information is also parsed into name and value pairs.

303

304 Same philosophy that was described above in request method can be applied to post data as  
305 well. Post data is parsed into name and value pairs and will be validated through a query string.

306

307 It is important to note that both the raw request and response are required because there are  
308 instances where the vulnerability and its probe contain a malformed header structure that cannot  
309 be parsed. Therefore, both the raw and parsed information will be provided in all parts of the  
310 specification.

311

```
312 <http-traversal>  
313   <request method="GET" connection="" host="172.16.50.31:80" request-uri="/"  
314   version="HTTP/1.0">
```

```

315 <raw>GET / HTTP/1.0 Connection: Close Host: 172.16.50.31
316 User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0) Pragma: no-
317 cache</raw>
318 <parsed>
319   <header name="Connection" value="Close"/>
320   <header name="Host" value="172.16.50.31"/>
321   <header name="User-Agent" value="Mozilla/4.0 (compatible; MSIE 5.01; Windows
322   NT 5.0)"/>
323   <header name="Pragma" value="no-cache"/>
324   <query value=""/>
325   <content value=""/>
326 </parsed>
327 </request>
328 <response>
329   <raw>HTTP/1.1 302 Object moved Server: Microsoft-IIS/5.0 Date: Tue, 10 Feb
330   2004 13:29:39 GMT Location: banklogin.asp?serviceName=
331   FreebankCaastAccess&templateName=prod_sel.forte&source=
332   Freebank&AD_REFERRING_URL= http://www.Freebank.com Connection:
333   Keep-Alive Content-Length: 251 Content-Type: text/html Cache-control: private
334   Set-Cookie: ASPSESSIONIDGGQQQUIU= GJABGOGAEBIONOCNAGGKLNLF;
335   path=/&head&title&Object moved&/title&/head&body&
336   body&h1&Object Moved&/h1&This object may be found &a
337   HREF="banklogin.asp?serviceName=FreebankCaastAccess&
338   templateName=prod_sel.forte&source=Freebank&
339   AD_REFERRING_URL= http://www.Freebank.com"&here&
340   /a&body&</raw>
341   <parsed>
342     <statusline value="HTTP/1.1 302 Object moved"/>
343     <header name="Server" value="Microsoft-IIS/5.0"/>
344     <header name="Date" value="Tue, 10 Feb 2004 13:29:39 GMT"/>
345     <header name="Location" value="banklogin.asp?serviceName=
346     FreebankCaastAccess&templateName=prod_sel.forte&source=
347     Freebank&AD_REFERRING_URL=http://www.Freebank.com"/>
348     <header name="Connection" value="Keep-Alive"/>
349     <header name="Content-Length" value="251"/>
350     <header name="Content-Type" value="text/html"/>
351     <header name="Cache-control" value="private"/>
352     <content>
353       <href uri="banklogin.asp?serviceName=FreebankCaastAccess&
354       templateName=prod_sel.forte&source=
355       Freebank&AD_REFERRING_URL=http://www.Freebank.com"
356       type="static" persistence="export"/>
357       <href uri="banklogin.asp?serviceName=FreebankCaastAccess&
358       templateName=prod_sel.forte&source=
359       Freebank&AD_REFERRING_URL=http://www.Freebank.com"
360       type="static" persistence="export"/>
361       <href uri="banklogin.asp" type="static" persistence="export"/>
362     </content>
363   </parsed>
364 </response>
365 </http-traversal>

```

366

## 367 2.3 Vulnerability Probe

368 The Vulnerability Probe is the second major section in the AVDL output. While the Traversal  
369 section maps the Web application and describes the requests and responses for each page of a  
370 Web application, the Vulnerability Probe section describes the vulnerabilities contained within the  
371 Web application.

372

373 The Vulnerability Probe is structured much like the Traversal. It is associated with a session and  
374 can contain many Containers each of which describes a single vulnerability of the Web  
375 application. In addition, a Vulnerability Probe can contain multiple Test Probes. For example, first  
376 test for general SQL injection then specific injection. Each Test Probe is contained within the  
377 Vulnerability Probe.

378

379 Continuing the example set forth previously, the Vulnerability Probe contains a header with the ID  
380 of the session that it is associated with, the target URL that contains the vulnerability, when the  
381 activity was started, and the vulnerability probe ID that is an identifier that is associated with the  
382 sequential order that this vulnerability was identified on the site.

383

```
384 <session id="vulnerability-session" target="http://172.16.50.31" session-start="2004-02-  
385 10T16:57:25">  
386   <vulnerability-probe time-stamp="2004-02-10T16:57:25">
```

387

### 388 **2.3.1 Vulnerability Probe Container**

389 Following this metadata information, the Vulnerability Probe contains both the raw request and  
390 response and the parsed request and response of the probe. Each Vulnerability Container  
391 contains one and only one vulnerability probe that includes one round-trip HTTP request to and  
392 response from the server. Like the Traversal Container, each Vulnerability Probe Container  
393 contains only one request/response pair. While each Vulnerability Probe Container contains only  
394 one request and response, a Session may contain many Vulnerability Probe Containers. In  
395 general, to complete a single round trip, a probe may encompass multiple protocols, each of  
396 which will contain its own request/response pair.

397

398 The probe contains a unique identifier within a single AVDL file and a time stamp to indicate when  
399 the vulnerability was found. It also contains a Test Probe that includes information that indicates  
400 how the vulnerability was found so that the test can be reproduced as necessary. It contains an  
401 identifier and a Test Script Reference. The Test Script Reference is a reference to the  
402 vulnerability test. This is the reference to reproduce the vulnerability. The Test Probe contains an  
403 HTTP Probe that includes the request method, the connection, host, request URI, and version of  
404 the protocol that was used. This is followed by the raw request and then the parsed request that  
405 was submitted by the Test Probe to identify the vulnerability. The request and response is parsed  
406 into header name and value pairs.

407

408 Within the standard, each request/response pair is represented in both raw and parsed form.  
409 Vulnerability Probe Containers are listed in chronological order. In addition, each container can  
410 have its own specific rules. These rules are also captured within the Vulnerability Probe  
411 Container.

412

413 It is important to note that both the raw request and response are required because there are  
414 instances where the vulnerability and its probe contain a malformed header structure that cannot  
415 be parsed. Therefore, both the raw and parsed information will be provided in all parts of the  
416 specification.

417

418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477

```
<test-probe>
  <http-probe>
    <request method="GET" connection="" host="172.16.50.31:80" request-uri=
"/banklogin.asp\" version="HTTP/1.0">
      <raw>GET /banklogin.asp HTTP/1.0 Referer: http://172.16.50.31:80/
Connection: Close Host: 172.16.50.31 User-Agent: Mozilla/4.0 (compatible; MSIE
5.01; Windows NT 5.0) Pragma: no-cache Translate: f Cookie:
ASPSESSIONIDGGQQUIU=GJABGOGAEBIONOCNAGGKLNLF;
CustomCookie=WebInspect</raw>
      <parsed>
        <header name="Referer" value="http://172.16.50.31:80/">
        <header name="Connection" value="Close"/>
        <header name="Host" value="172.16.50.31"/>
        <header name="User-Agent" value="Mozilla/4.0 (compatible; MSIE 5.01;
Windows NT 5.0)"/>
        <header name="Translate" value="f"/>
        <query value=""/>
        <content value=""/>
      </parsed>
    </request>
    <response>
      <raw>HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Tue, 10 Feb 2004 13:32:06 GMT
Content-Type: application/octet-stream
Content-Length: 5353
&lt;%
response.cookies("userid") = "" %&lt;

      </raw>
      <parsed>
        <statusline value="HTTP/1.1 200 OK"/>
        <header name="Server" value="Microsoft-IIS/5.0"/>
        <header name="Date" value="Tue, 10 Feb 2004 13:32:06 GMT"/>
        <header name="Content-Type" value="application/octet-stream"/>
        <header name="Content-Length" value="5353"/>
        <content>
          <href uri="images/freebank-logo2.gif" type="static"
persistence="export"/>
          <href uri="images/lock.gif" type="static" persistence="export"/>
          <href uri="images/customer-login.gif" type="static"
persistence="export"/>
          <href uri="images/financial-planning.gif" type="static"
persistence="export"/>
          <href uri="images/services.gif" type="static" persistence="export"/>
          <href uri="images/your-accounts.gif" type="static"
persistence="export"/>
          <href uri="redirect1/redirect1.asp" type="static" persistence="export"/>
          <href uri="pindex.asp" type="static" persistence="export"/>
          <href uri="bookstore/java/default.htm" type="static"
persistence="export"/>
          <href uri="login1.asp" type="static" persistence="export"/>
          <href uri="rootlogin.asp" type="static" persistence="export"/>
        </content>
      </parsed>
    </response>
  </http-probe>
</test-probe>
```

478

## 479 **2.3.2 Vulnerability Properties**

480 The Vulnerability Properties describe the vulnerability and are intended for use in the “human”  
481 interface display. For this version of the standard, English will be used to complete the properties.  
482 However, it is envisioned that other languages will be supported in future versions. The  
483 Properties of the vulnerability contain

- 484 • Summary - a brief description of the vulnerability
- 485 • Description - a detailed description of the vulnerability
- 486 • Classification - a unique identifier for the vulnerability
- 487 • Datum - metadata about the vulnerability
- 488 • History - the version of the vulnerability that was used

489

```
490 <vulnerability-description title="IIS Translate:f Source Code Disclosure">
```

491

492 Subsequent sections will provide more detail to the Vulnerability properties.

### 493 **2.3.2.1 Summary**

494 The Summary provides a brief description of the vulnerability. It should contain one or two  
495 sentences describing the vulnerability and its purpose. The Summary is not intended to provide  
496 detailed information, but is intended to be brief. It is recommended that this information provide  
497 overall context for the vulnerability.

498

499 The following is an example of the Summary for the Translate f vulnerability:

500

```
501 <summary>A vulnerability in IIS allows remote attackers to view the source of offered server  
502 side scripts supported by IIS (such as ASP, ASA, HTR, etc.) by using malformed "Translate: f"  
503 header.</summary>
```

504

### 505 **2.3.2.2 Description**

506 The Description is a detailed explanation of the vulnerability. It should describe what the  
507 vulnerability is, what systems are susceptible to it, the history of the vulnerability, and any other  
508 relevant information regarding the vulnerability. The description is displayed in paragraph form as  
509 shown in the following example:

510

```
511 <description>A vulnerability in IIS allows remote attackers to view the source of offered server  
512 side scripts supported by IIS (such as ASP, ASA, HTR, etc.). This vulnerability is very dangerous  
513 since a lot of sensitive information is kept in these files, as programmers often rely on the fact  
514 that the source code is hidden from the user. The vulnerability involves sending a special header  
515 with 'Translate: f' at the end of it, and then a trailing slash '/' appended to the end of the URL. It  
516 cannot be exploited by the standard browsers, but an exploit code below enables to test for this  
517 problem.</description>
```

518

### 519 **2.3.2.3 Classification**

520 The Classification of the vulnerability is its unique global name. This name is expected to be  
521 developed by other standards bodies. The classification also includes a severity rating that  
522 indicates, on a scale from 1to100, how important the vulnerability is. Vulnerabilities with a score  
523 of 100 are the most critical while those of a score of 1 are more informational.

### 524 **2.3.3 Vulnerability Specific**

525 Information contained within this section of the output includes the specific information about how  
526 the vulnerability was discovered. This includes information regarding the target application, the  
527 test attack, and a description of the attack. The following subsections describe each portion of the  
528 vulnerability target.

529

#### 530 **2.3.3.1 Test**

531 The Test is an important aspect of the output because it describes the specific test script that was  
532 used to identify the vulnerability on the web server. It is the test that was used to scan the target  
533 web application. The Test includes an identifier and a reference to the target application that was  
534 attacked. The following example displays these values:

535

```
<test-script id="test-script-1">
```

537

#### 538 **2.3.3.2 Test description**

539 The Test Description contains information about the specific vulnerability, such as when and how  
540 it was detected. It also includes the request and response (in raw form) that was used to detect  
541 this vulnerability. This will allow recipients of the output to reproduce the vulnerability.

542

543 The raw request is broken down in this portion of the standard to provide more details of the  
544 attack. In this example request, the two attack components are Translate: f and GET ending in  
545 backslash. All the details are listed here. The response includes the expected result from the  
546 server. If the response returns the expected result, then the vulnerability has been confirmed. The  
547 following example depicts a specific attack test:

548

```
549 <declare name="path" type="string"/>  
550 <declare name="protocol" type="string" default="HTTP/1.1"/>  
551 <declare name="host" type="host"/>  
552 <declare name="port" type="integer" default="80"/>  
553 <sequence>  
554 <http-transaction>  
555 <request>GET &lt;var name="path"/&gt; &lt;var name="protocol"/&gt; Referer:  
556 http://&lt;var name="host"/&gt;:&lt;var name="port"/&gt;/  
557 Connection: Close  
558 Host: &lt;var name="host"/&gt;  
559 User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)  
560 Pragma: no-cache  
561 Translate: f  
562 Cookie: ASPSESSIONIDGGQQUIU=GJABGOGAEBIONOCNAGGKNLNF;  
563 CustomCookie=WebInspect</request>  
564 <response>
```

565  
566  
567  
568

```
<expect status-code="200" reason-phrase="OK"/>
</response>
</http-transaction>
</sequence>
```

569

### 570 2.3.3.3 Remediation

571 Remediation is the recommended action to close the vulnerability. It includes an identifier for the  
572 remedy, a description, and the vendor responsible for creating the remedy. The action code is  
573 vendor specific to the vendor specified by the Vendor field. In addition, it includes an open block  
574 that allows for machine-readable code. This may include code for the remediation software to  
575 download the patch to fix the vulnerability.

576

577  
578  
579

```
<recommendation>
<patch name="Microsoft patch Q256888_W2K_SP1_x86_en" lang="english" test-
ref="test-1">
<description>Microsoft has released a patch which eliminates this vulnerability.
</description>
<vendor name="Microsoft" />
<patch-source href="http://download.microsoft.com/download/win2000platform/Patch
/Q256888/NT5/EN-US/Q256888_W2K_SP1_x86_en.EXE" patch-ref="Q256888_W2K_
SP1_x86_en" />
<remediation vulnID="02134" language="VBScript" modDate=
"030911131212" vendor="Citadel" actionhref=
"http://vendor.remediation.com/library/q25688.vb" actionCode="REM
Copyright 2003, Citadel Security Software, Inc. All Rights Reserved. All product names
are trademarks or registered trademarks of their respective owners. Specifications
subject to change without notice. REM Script Generated Automatically by skey at
9/10/2003 2:04:30 PM Option Explicit HercClient.SetScriptReturnCode( 5 ) REM Failure
Dim sVersion, sFull, sSP, bPassed bPassed = true If bPassed = true Then If
HercClient.IsWindowsXP() = True then If HercClient.WindowsCSDVersion > Service
Pack 1 Then bPassed = True Else bPassed = False End If End If End If" />
</patch name>
</recommendation>
```

588

599

---

600

## Appendix A. Acknowledgments

601 The AVDL Technical Committee would like to acknowledge earlier efforts in promotion of  
602 application vulnerabilities and standardization of their representation and interchange. Their work  
603 inspired many ideas incorporated into the AVDL standard.

604 Open Vulnerability Assessment Language developed at the Mitre Corporation “is the common  
605 language for security experts to discuss and agree upon technical details about how to check for  
606 the presence of a vulnerability on a computer system”. Using SQL, OVAL queries are based on  
607 broadly recognized Common Vulnerabilities and Exposures (CVE) database and by “specifying  
608 logical conditions on the values of system characteristics and configuration attributes, OVAL  
609 queries characterize exactly which systems are susceptible to a given vulnerability.”

610 VulnXML developed by a OWASP team led by Mark Curphey “could be used by automated  
611 assessment tools to test for known security issues”. Closely related and also developed at  
612 OWASP was Application Security Attack Components or ASAC which “is a basic classification  
613 scheme of web application security issues. The aim of this project was to create a common  
614 language and a consensus understanding among the industry to describe the same issue in the  
615 same way.” Their work continues at OASIS Web Application Security TC.

616

617

---

## Appendix B. Revision History

Rev	Date	By Whom	What
wd-01	2004-01-08	Kevin Heineman	Version 1.0
wd-02	2004-01-18	Carl Banzhof	Added provider attribute to root block
Wd-03	2004-03-08	Kevin Heineman	Modifications made from Working Draft comments.
Wd-04	2004-03-11	Kevin Heineman	Simplified the example,
Wd-05	2004-06-08	Kevin Heineman	Updated title and footer to reflect OASIS standard.

618

---

619

## Appendix C. Notices

620 OASIS takes no position regarding the validity or scope of any intellectual property or other rights  
621 that might be claimed to pertain to the implementation or use of the technology described in this  
622 document or the extent to which any license under such rights might or might not be available;  
623 neither does it represent that it has made any effort to identify any such rights. Information on  
624 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS  
625 website. Copies of claims of rights made available for publication and any assurances of licenses  
626 to be made available, or the result of an attempt made to obtain a general license or permission  
627 for the use of such proprietary rights by implementers or users of this specification, can be  
628 obtained from the OASIS Executive Director.

629

630 OASIS invites any interested party to bring to its attention any copyrights, patents or patent  
631 applications, or other proprietary rights that may cover technology that may be required to  
632 implement this specification. Please address the information to the OASIS Executive Director.

633

634 Copyright © OASIS Open 2004. *All Rights Reserved.*

635

636 This document and translations of it may be copied and furnished to others, and derivative works  
637 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,  
638 published and distributed, in whole or in part, without restriction of any kind, provided that the  
639 above copyright notice and this paragraph are included on all such copies and derivative works.  
640 However, this document itself does not be modified in any way, such as by removing the  
641 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS  
642 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual  
643 Property Rights document must be followed, or as required to translate it into languages other  
644 than English.

645

646 The limited permissions granted above are perpetual and will not be revoked by OASIS or its  
647 successors or assigns.

648

649 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
650 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO  
651 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE  
652 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A  
653 PARTICULAR PURPOSE.