



Application Vulnerability Description Language

Working Draft 01, 15 January 2004

Document identifier:

AVDL Specification - 01

Location:

<http://TBD>

Editor:

Jan Bialkowski, NetContinuum, jan@netcontinuum.com
Kevin Heineman, SPI Dynamics, kheineman@spidynamics.com

Contributors:

Carl Banzhof, Citadel
John Diaz, Lawrence Livermore National Laboratory
Johan Strandberg, NetContinuum
Srinivas Mantripragada, NetContinuum
Caleb Sima, SPI Dynamics

Participants:

Jeremy Poteet, Individual
Lauren Davis, Johns Hopkins University Applied Physics Laboratory
Andrew Buttner, Mitre Corporation
Gerhard Eschelbeck, Qualys
Jared Karro, Bank of America
Montgomery-Recht Evan, Booz Allen Hamilton
Ajay Gummadi, Individual
Yen-Ming Chen, Individual
Brian Cohen, SPI Dynamics, Inc.
John Milciunas, SPI Dynamics, Inc.
Matthew Snyder, Bank of America
Chung-Ming Ou, Chunghwa Telecom Laboratories
Anton Chuvakin, Individual
Nasseam Elkarra, Individual
Roger Alexander, Individual
J. Wittbold, Mitre Corporation
Lluis Mora, Sentryware

Abstract:

This specification describes a standard XML format that allows entities (such as applications, organizations, or institutes) to communicate information regarding web application vulnerabilities. Simply said, Application Vulnerability Description Language (AVDL) is a security interoperability

40 standard for creating a uniform method of describing application security vulnerabilities using
41 XML.

42

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

54

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

59

60 **Status:**

61 This document is the AVDL Technical Committee Specification. Please send comments to the
62 editors.

63

64 Committee members should send comments on this specification to avdl@lists.oasis-open.org.
65 Others should subscribe to and send comments to avdl-comment@lists.oasis-open.org. To
66 subscribe, send an email message to avdl-comment-request@lists.oasis-open.org with the word
67 "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 the
71 Intellectual Property Rights section of the AVDL Technical Committee (AVDL TC) web page
72 (<http://www.oasis-open.org/committees/avdl/ipr.php>).

73

74 **Eratta:**

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

77 **Table of Contents**

78 Introduction 4
79 1.1 Notations and Terminology 5
80 1.1.1 Notations 5
81 1.1.2 Terminology 5
82 1.2 Requirements 6
83 1.3 Out of Scope 6
84 2 AVDL Output 8
85 2.1 AVDL File Root 8
86 2.2 Traversal 8
87 2.2.1 Post Data Container with Parameter Validations 9
88 2.3 Vulnerability Probe Example 11
89 2.3.1 Vulnerability Properties 13
90 2.3.2 Vulnerability Specific 15
91 Appendix A. XML Example 18
92 Appendix B. Acknowledgments 18
93 Appendix C. Revision History 27
94 Appendix D. Notices 28
95

96 Introduction

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

107

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

120

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

126

127 Vulnerability information may include:

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

133

134 AVDL is capable of describing either type of information.

135

136 1.1 Notations and Terminology

137 1.1.1 Notations

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

141

142 1.1.2 Terminology

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

- 179 • **Summary** – This descriptor defines a short summary of the vulnerability within the Test
180 Probe.
- 181 • **Target ID** – This descriptor classifies the target operating system that is associated with the
182 vulnerability contained within the Test Probe.
- 183 • **Target Ref** – This descriptor identifies the operating system of the test target.
- 184 • **Test Probe** – This is a container of the session that identified the vulnerability. The Probe
185 contains both the raw request and raw response as well as parsed request and parsed
186 response.
- 187 • **Test Probe ID** – This identifier classifies the specific test that produced a vulnerability.
- 188 • **Test Script ID** – This descriptor identifies the test that was conducted as part of the Test
189 Probe to identify the vulnerability. A Test Probe may contain one to many Test Scripts.
- 190 • **Traversal Step ID** – A traversal is the sum of a request to a web server and a response from
191 the web server. Each Traversal Step is identified with a unique identifier. The Traversal Step
192 contains both the raw and parsed content of the request and response.
- 193 • **Vulnerability Description Title** – This descriptor defines the vulnerability within the Test
194 Probe.
- 195 • **Vulnerability Probe** – This is a container for the Test Probes and may contain one to many
196 Test Probes.
- 197 • **Vulnerability Probe ID** – This identifier classifies the probes that were used to identify
198 vulnerabilities. The term “Probe” is used since the application originating the data is generic
199 (e.g., assessment, protection, remediation, event correlation).
200

201 1.2 Requirements

202 The Application Vulnerability Description Language uses XML to support communication between
203 applications that exchange information about web application vulnerabilities. Specifically the
204 specification includes two major sections: Traversal and Vulnerability Probe.
205

206 The Traversal is a mapping of the structure of the site. Its purpose is to fully enumerate the web
207 application. The Traversal is populated by assessment products to map the application and
208 create a baseline of the site. It describes the requests and responses that were made to the
209 server and the pages that were displayed as a result of the requests.
210

211 The Vulnerability Probe is a description of a vulnerability. It includes information about the
212 vulnerability as well as how the vulnerability was found and, when possible, how it can be fixed.
213

214 1.3 Out of Scope

215 AVDL has been developed to describe web application vulnerabilities. It is not intended to be
216 used to describe other types of vulnerabilities. This includes (but is not limited to) server,
217 operating system, TCP related attacks, or other network layer issues. While vulnerabilities of
218 these types may also fit within the AVDL model, the standard was not specifically developed for
219 these types of vulnerabilities.
220

221 AVDL is not intended to carry any information about authentication or access control. These
222 issues are covered by SAML and XACML.
223
224 Version 1.0 of the standard is specific to English language output. Future versions of the standard
225 are anticipated to address or accommodate other languages.
226
227 Encapsulating well-defined behavior of the target application within the standard is not within the
228 scope of AVDL version 1.0. Well-defined behavior is specific information relating to how the web
229 application works. For example, valid values for a page as well as the behavior of the application
230 with regards to invalid values. Discrepancies to this normal behavior would be identified as
231 vulnerabilities. Future versions of the standard may address this issue.
232
233 A complete catalog of the potential vulnerabilities is not included in the specification. The
234 standard will not contain any descriptors that contain any vulnerability storage containers. This
235 includes either content or a list of identifiers (such as CVE).
236
237 This version of the AVDL standard addresses only web application vulnerabilities. Future versions
238 of the standard may incorporate the output from other vulnerability scanners that are not web-
239 based such as ISS and other probes.

240 2 AVDL Output

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

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

248

249 2.1 AVDL File Root

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

254

```
255 - <avdl version="0.1-2003-09-27" provider="SPI"  
256 xmlns="urn:oasis:names:tc:avdl:0.0:mailto:avdl@oasis-open.org?:avdl:2003-09-27:a"  
257 xmlns:xhtml="http://www.w3.org/1999/xhtml"  
258 xmlns:avdl="urn:oasis:names:tc:avdl:0.0:names:mailto:avdl@oasis-open.org?:2003-09-"  
259 xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

260

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

267

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

271

272 2.2 Traversal

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

279

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

282 traversal step ID (a number designating this session in the ordered sequence of nodes visited
283 during the crawl).). It also contains the raw request and response and the parsed request and
284 response.

285

```
286 - <session id="session-1" target="http://www.example.com/" session-start="2003-09-  
287 27T10:35:49">  
288 - <traversal-step id="step-1234" time-stamp="23.124" sequence-number="1234"  
289 uri="http://www.example.com:80/plink.asp?a=3&c=xyz">  
290 - <http-traversal>
```

291

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

295

296 2.2.1 Traversal Container

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

302

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

306

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

309

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

316

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

319

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

324

```
325 - <request method="GET" connection="proxy.example.com:8080" host="www.example.com:80"  
326 request-uri="/plink.asp?a=3&c=xyz" version="HTTP/1.0">  
327 - <raw>
```

```

328 GET /plink.asp?a=3&c=xyz HTTP/1.0
329 <eol />
330 Referer: http://www.example.com:80/pindex.asp
331 <eol />
332 Connection: Close
333 <eol />
334 Host: www.example.com:80
335 <eol />
336 User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
337 <eol />
338 Pragma: no-cache
339 <eol />
340 Cookie: ASPSESSIONIDSQBRQDDT=MCKFENJJCFCCKDDPANEKECMK; sessionId=; state=;
341 username=; userid=; CustomCookie=WebInspect
342 <eol />
343 </raw>
344 - <parsed>
345 <header name="Cookie" value="ASPSESSIONIDSQBRQDDT=MCKFENJJCFCCKDDPANEKECMK;
346 sessionId=; state=; username=; userid=; CustomCookie=WebInspect" />
347 <header name="Referer" value="http://www.example.com:80/pindex.asp" />
348 <header name="User-Agent" value="Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)"
349 />
350 <header name="Pragma" value="no-cache" />
351 - <query value="a=3&c=xyz">
352 - <parameter name="a" value="3">
353 <test type="int" />
354 <test greater-or-equals="0" />
355 <test less-or-equals="123456" />
356 </parameter>
357 - <parameter name="c" value="xyz">
358 <test max-length="3" />
359 </parameter>
360 </query>
361 - <content>
362 </content>
363 </parsed>
364 </request>
365 - <response>
366 - <raw>
367 HTTP/1.1 200 OK
368 <eol />
369 Server: Microsoft-IIS/5.0
370 <eol />
371 Date: Fri, 01 Aug 2003 02:28:12 GMT
372 <eol />
373 X-Powered-By: ASP.NET
374 <eol />
375 Connection: Keep-Alive
376 <eol />
377 Content-Length: 167
378 <eol />
379 Content-Type: text/html
380 <eol />
381 Cache-Control: private
382 <eol />
383 <eol />
384 <eol />
385 <html>
386 <eol />
387 <body>

```

```

388 <eol />
389 <eol />
390 <a href="pindex.asp">Click here to return to index</a>
391 <eol />
392 <eol />
393 </body>
394 <eol />
395 </html>
396 <eol />
397 </raw>
398 - <parsed base="http://www.example.com:80/">
399 <statusline value="HTTP/1.1 200 OK" />
400 <statusline name="protocol" value="HTTP/1.1" />
401 <statusline name="status-code" value="200" />
402 <statusline name="reason-phrase" value="OK" />
403 <header name="Server" value="Microsoft-IIS/5.0" />
404 <header name="Date" value="Fri, 01 Aug 2003 02:28:12 GMT" />
405 <header name="X-Powered-By" value="ASP.NET" />
406 <header name="Connection" value="Keep-Alive" />
407 <header name="Content-Length" value="167" />
408 - <header name="Content-Type" value="text/html">
409 <test equals="text/html" ignore-case="true" />
410 </header>
411 <header name="Cache-Control" value="private" />
412 - <content>
413 <href uri="pindex.asp" type="static" persistence="export" />
414 </content>
415 </parsed>
416 </response>
417 </http-traversal>
418 </traversal-step>
419 </session>

```

420

421 2.3 Vulnerability Probe

422 The Vulnerability Probe is the second major section in the AVDL output. While the Traversal
423 section maps the Web application and describes the requests and responses for each page of a
424 Web application, the Vulnerability Probe section describes the vulnerabilities contained within the
425 Web application.

426

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

432

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

437

```

438 - <session id="session-2" target="http://www.example.com/" session-start="2003-09-
439 27T10:35:49">

```

440 - <vulnerability-probe id="probe-1234" time-stamp="23.124">

441

442 2.3.1 Vulnerability Probe Container

443 Following this metadata information, the Vulnerability Probe contains both the raw request and
444 response and the parsed request and response of the probe. Each Vulnerability Container
445 contains one and only one vulnerability probe that includes one round-trip HTTP request to and
446 response from the server. Like the Traversal Container, each Vulnerability Probe Container
447 contains only one request/response pair. While each Vulnerability Probe Container contains only
448 one request and response, a Session may contain many Vulnerability Probe Containers. In
449 general, to complete a single round trip, a probe may encompass multiple protocols, each of
450 which will contain its own request/response pair.

451

452 The probe contains a unique identifier within a single AVDL file and a time stamp to indicate when
453 the vulnerability was found. It also contains a Test Probe that includes information that indicates
454 how the vulnerability was found so that the test can be reproduced as necessary. It contains an
455 identifier and a Test Script Reference. The Test Script Reference is a reference to the
456 vulnerability test. This is the reference to reproduce the vulnerability. The Test Probe contains an
457 HTTP Probe that includes the request method, the connection, host, request URI, and version of
458 the protocol that was used. This is followed by the raw request and then the parsed request that
459 was submitted by the Test Probe to identify the vulnerability. The request and response is parsed
460 into header name and value pairs.

461

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

466

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

471

```
472 - <test-probe id="test-probe-1" test-script-ref="test-1">
473 - <http-probe>
474 - <request method="GET" connection="proxy.example.com:8080" host="www.example.com:80"
475 request-uri="/login.asp\" version="HTTP/1.1">
476 - <raw>
477   GET /login.asp\ HTTP/1.1
478   <eol />
479   Host: example.com:80
480   <eol />
481   User-Agent: SensePostData
482   <eol />
483   Content-Type: application/x-www-form-urlencoded
484   <eol />
485   Translate: f
486   <eol />
487 </raw>
488 - <parsed>
489   <header name="User-Agent" value="SensePostData" />
```

```

490 <header name="Content-Type" value="application/x-www-form-urlencoded" />
491 </parsed>
492 </request>
493 - <response>
494 - <raw>
495 HTTP/1.1 200 OK
496 <eol />
497 Server: Microsoft-IIS/5.0
498 <eol />
499 Date: Fri, 01 Aug 2003 02:28:12 GMT
500 <eol />
501 X-Powered-By: ASP.NET
502 <eol />
503 Connection: Keep-Alive
504 <eol />
505 Content-Length: 167
506 <eol />
507 Content-Type: text/html
508 <eol />
509 </raw>
510 - <parsed>
511 <statusline name="status-code" value="200" />
512 <header name="Server" value="Microsoft-IIS/5.0" />
513 <header name="Date" value="Fri, 01 Aug 2003 02:28:12 GMT" />
514 <header name="X-Powered-By" value="ASP.NET" />
515 <header name="Connection" value="Keep-Alive" />
516 <header name="Content-Length" value="167" />
517 <header name="Content-Type" value="text/html" />
518 </parsed>
519 </response>
520 </http-probe>
521 </test-probe>
522 - <vulnerability-description title="IIS Translate:f Source Code Disclosure" version="2003-09-
523 27JS-1" id="vulnerability-1">
524

```

525 2.3.2 Vulnerability Properties

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

- 530 • Summary - a brief description of the vulnerability
- 531 • Description - a detailed description of the vulnerability
- 532 • Classification - a unique identifier for the vulnerability
- 533 • Datum - metadata about the vulnerability
- 534 • History - the version of the vulnerability that was used

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

536 2.3.2.1 Summary

537 The Summary provides a brief description of the vulnerability. It should contain one or two
538 sentences describing the vulnerability and its purpose. The Summary is not intended to provide

539 detailed information, but is intended to be brief. It is recommended that this information provide
540 overall context for the vulnerability.

541

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

543

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

547

548 2.3.2.2 Description

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

553

```
554 - <description xml:lang="en">  
555   A vulnerability in IIS allows remote attackers to view the source of offered server side  
556   scripts supported by IIS (such as ASP, ASA, HTR, etc.).  
557   <xhtml:p />  
558   This vulnerability is very dangerous since a lot of sensitive information is kept in these  
559   files, as programmers often rely on the fact that the source code is hidden from the user.  
560   The vulnerability involves sending a special header with 'Translate: f' at the end of it, and  
561   then a trailing back-slash '\' appended to the end of the URL. It cannot be exploited by the  
562   standard browsers, but an exploit code below enables to test for this problem.  
563   <xhtml:p />  
564   WebDAV implemented in Windows 2000 and Office 2000 (including FrontPage 2000 and  
565   FrontPage 2000 Server extensions) is the source of Translate:f problem.  
566   <xhtml:p />  
567   When someone makes request for ASP/ASA (or any other scriptable page) and adds  
568   "Translate: f" into headers of HTTP GET request (headers are not part of the URL, they are  
569   part of the raw HTTP request), there is a serious security bug in Windows 2000 (when  
570   unpatched by SP1) that in return gives complete ASP/ASA code instead of processed file.  
571   It's necessary to add a trailing back-slash "\" to end of requested URL to make this work.  
572   <xhtml:p />  
573   "Translate:f" is legitimate header for WebDAV, it is used as it should be - adding this to  
574   HTTP GET is a signal for the WebDAV component to return the source code of the  
575   requested file and bypass processing. It is used in FrontPage2000 and any WebDAV  
576   compatible client to get a file for editing. It has to be accompanied by some other  
577   information, which should prevent unauthorized users from viewing the source.  
578   Unfortunately, a coding problem makes it possible to retrieve those files by simply adding  
579   "Translate:f" in the header, and placing "\" at end of request to the HTTP GET.  
580   <xhtml:p />  
581   It is a Windows 2000 bug, but because of FrontPage Server Extensions 2000 can be  
582   installed even on IIS 4.0 sites, it also affectes IIS 4.0. Many IIS 4.0 sites will exhibit the  
583   "Translate: f" bug when web files are stored on a shared (network) directory, this  
584   vulnerability has been fixed in the past (see our previous article: Patch Available for the  
585   Virtualized UNC Share Vulnerability).  
586 </description>
```

587

588 2.3.2.3 Classification

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

593

594 2.3.2.4 Datum

595 Datum is metadata regarding the vulnerability. It includes information such as the date the
596 vulnerability was found and who found it. The entity that is listed in the Datum is also the entity
597 that created the other information about the vulnerability. Any updates to the vulnerability content
598 will also be listed in the Datum as well as the party who was responsible for making the changes.
599 The following example illustrates the type of information that is included in the Datum.

600

```
601 <datum name="avdl:date-found" type="date" value="2000-06-05" />  
602 <datum name="avdl:found-by" type="string" value="SecurITeam" />
```

603

604 2.3.2.5 History

605 In some cases, multiple versions of the vulnerability may be available. The history section clearly
606 states which version is being referenced and the version history of the vulnerability. The following
607 example shows a sample output for this section.

608

```
609 <history uri="http://www.oasis-  
610 open.org/apps/org/workgroup/avdl/download.php/2927/avdlstruct1.xml" />
```

611

612 2.3.3 Vulnerability Specific

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

617

618 2.3.3.1 Target

619 The Target contains information regarding the server that was attacked. The information includes
620 an identifier for the target system, the operating system the server was using, the hardware
621 running the server, the name and version of the web server, and the protocol used. This is shown
622 in the following example:

623

```
624 - <target id="target-win2k">  
625   <os name="Microsoft Windows 2000 Version 2.1 Service Pack 3" />  
626   <arch name="Intel Pentium III" />  
627   <webserver name="IIS 4.0" />  
628   <webserver name="WebDAV" />  
629   <protocol name="HTTP 1.0" />  
630 </target>
```

631

632 2.3.3.2 Test

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

637

```
638 - <test-script id="test-1" target-ref="target-win2k">
```

639

640 2.3.3.3 Test description

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

644

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

650

```
651 <declare name="proxy-host" type="host" />
652 <declare name="proxy-port" type="integer" default="8080" />
653 <declare name="host" type="host" />
654 <declare name="port" type="integer" default="80" />
655 <declare name="path" type="string" />
656 <declare name="protocol" type="string" default="HTTP/1.1" />
657 - <sequence repeat="1">
658 - <http-transaction>
659 - <request>
660   GET
661   <space />
662   /
663   <var name="path" />
664   <attack comment="GET ending in backslash">\</attack>
665   <space />
666   <var name="protocol" />
667   <eol />
668   Host:
669   <space />
670   <var name="host" />
671   :
672   <var name="port" />
673   <eol />
674   User-Agent:
675   <space />
676   SensePostData
677   <eol />
678   Content-Type:
679   <space />
680   application/x-www-form-urlencoded
```



```

681 <eol />
682 <attack comment="Should have required additional info to return source">Translate:
683 f</attack>
684 <eol />
685 </request>
686 - <response>
687 - <expect status-code="200" reason-phrase="OK">
688 <match-header name="Content-Type" value="application/octet-stream" />
689 </expect>
690 </response>
691 </http-transaction>
692 </sequence>
693 </test-script>

```

694

695 2.3.3.4 Remediation

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

701

```

702 - <recommendation>
703 - <patch name="Microsoft patch Q256888_W2K_SP1_x86_en" lang="english" test-ref="test-1">
704 <description>Microsoft has released a patch which eliminates this
705 vulnerability.</description>
706 <vendor name="Microsoft" />
707 <patch-source
708 href="http://download.microsoft.com/download/win2000platform/Patch/Q256888/NT5/EN-
709 US/Q256888_W2K_SP1_x86_en.EXE" patch-ref="Q256888_W2K_SP1_x86_en" />
710 <remediation vulnID="02134" language="VBScript" modDate="030911131212"
711 vendor="Citadel" actionhref="http://vendor.remediation.com/library/q25688.vb"
712 actionCode="REM Copyright 2003, Citadel Security Software, Inc. All Rights Reserved.
713 All product names are trademarks or registered trademarks of their respective owners.
714 Specifications subject to change without notice. REM Script Generated Automatically
715 by skey at 9/10/2003 2:04:30 PM Option Explicit HercClient.SetScriptReturnCode( 5 )
716 REM Failure Dim sVersion, sFull, sSP, bPassed bPassed = true If bPassed = true Then
717 If HercClient.IsWindowsXP() = True then If HercClient.WindowsCSDVersion > Service
718 Pack 1 Then bPassed = True Else bPassed = False End If End If End If" />
719 </patch>
720 <user-description />
721 </recommendation>
722 </vulnerability-description>
723 </vulnerability-probe>
724 </session>
725 </avdl>

```

726

727

728

Appendix A. XML Example

729

This section contains a full example of an AVDL output for the Translate f vulnerability.

730

731

```
<?xml version="1.0" encoding="UTF-8" ?>
```

732

```
- <!--
```

733

```
=====
```

734

```
Copyright © OASIS Open (2003). All Rights Reserved.
```

735

736

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

748

749

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

750

751

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

757

```
=====
```

759

```
- <!--
```

760

```
=== root of AVDL file ===
```

761

```
- <avdl version="0.1-2003-09-27" provider="SPI"
```

762

```
xmlns="urn:oasis:names:tc:avdl:0.0:mailto:avdl@oasis-open.org?:avdl:2003-09-27:a"
```

763

```
xmlns:xhtml="http://www.w3.org/1999/xhtml"
```

764

```
xmlns:avdl="urn:oasis:names:tc:avdl:0.0:names:mailto:avdl@oasis-open.org?:2003-09-27"
xmlns:XS="http://www.w3.org/2001/XMLSchema">
```

765

766

```
- <!--
```

767

```
=====
```

768

```
- <!--
```

769

```
==
```

770

```
- <!--
```

771

```
== traversal example ==
```

772

```
- <!--
```

773

```
==
```

774

```
- <!--
```

775

```
=====
```

776

```
- <session id="session-1" target="http://www.example.com/" session-start="2003-09-27T10:35:49">
```

777

778

```
- <traversal-step id="step-1234" time-stamp="23.124" sequence-number="1234"
```

779

```
uri="http://www.example.com:80/plink.asp?a=3&c=xyz">
```

780

```
- <http-traversal>
```

781

```
- <request method="GET" connection="proxy.example.com:8080"
```

782

```
host="www.example.com:80" request-uri="/plink.asp?a=3&c=xyz" version="HTTP/1.0">
```

```

783 - <raw>
784 GET /plink.asp?a=3&c=xyz HTTP/1.0
785 <eol />
786 Referer: http://www.example.com:80/pindex.asp
787 <eol />
788 Connection: Close
789 <eol />
790 Host: www.example.com:80
791 <eol />
792 User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
793 <eol />
794 Pragma: no-cache
795 <eol />
796 Cookie: ASPSESSIONIDSQBRQDDT=MCKFENJCJCFCKDDPANEKECMK;
797 sessionId=; state=; username=; userid=; CustomCookie=WebInspect
798 <eol />
799 </raw>
800 - <parsed>
801 <header name="Cookie" value="ASPSESSIONIDSQBRQDDT=MCKFENJCJCFCKDDPANEKECMK;
802 sessionId=; state=; username=; userid=; CustomCookie=WebInspect" />
803 <header name="Referer" value="http://www.example.com:80/pindex.asp" />
804 <header name="User-Agent" value="Mozilla/4.0 (compatible; MSIE 5.01; Windows NT
805 5.0)" />
806 <header name="Pragma" value="no-cache" />
807 - <query value="a=3&c=xyz">
808 - <parameter name="a" value="3">
809 <test type="int" />
810 <test greater-or-equals="0" />
811 <test less-or-equals="123456" />
812 </parameter>
813 - <parameter name="c" value="xyz">
814 <test max-length="3" />
815 </parameter>
816 </query>
817 - <content>
818 - <!--
819 post data container with parameter validations
820 </content>
821 </parsed>
822 </request>
823 - <response>
824 - <raw>
825 HTTP/1.1 200 OK
826 <eol />
827 Server: Microsoft-IIS/5.0
828 <eol />
829 Date: Fri, 01 Aug 2003 02:28:12 GMT
830 <eol />
831 X-Powered-By: ASP.NET
832 <eol />

```

```

833 Connection: Keep-Alive
834 <eol />
835 Content-Length: 167
836 <eol />
837 Content-Type: text/html
838 <eol />
839 Cache-Control: private
840 <eol />
841 <eol />
842 <eol />
843 <html>
844 <eol />
845 <body>
846 <eol />
847 <eol />
848 <a href="pindex.asp">Click here to return to index</a>
849 <eol />
850 <eol />
851 </body>
852 <eol />
853 </html>
854 <eol />
855 </raw>
856 - <parsed base="http://www.example.com:80/">
857 <statusline value="HTTP/1.1 200 OK" />
858 <statusline name="protocol" value="HTTP/1.1" />
859 <statusline name="status-code" value="200" />
860 <statusline name="reason-phrase" value="OK" />
861 <header name="Server" value="Microsoft-IIS/5.0" />
862 <header name="Date" value="Fri, 01 Aug 2003 02:28:12 GMT" />
863 <header name="X-Powered-By" value="ASP.NET" />
864 <header name="Connection" value="Keep-Alive" />
865 <header name="Content-Length" value="167" />
866 - <header name="Content-Type" value="text/html">
867 <test equals="text/html" ignore-case="true" />
868 </header>
869 <header name="Cache-Control" value="private" />
870 - <content>
871 <href uri="pindex.asp" type="static" persistence="export" />
872 </content>
873 </parsed>
874 </response>
875 </http-traversal>
876 </traversal-step>
877 </session>
878 - <!--
879 =====
880 - <!--
881 ==
882 - <!--

```

```

883 ==          vulnerability-probe example          ==
884 - <!--
885 ==
886 - <!--
887 =====
888 - <session id="session-2" target="http://www.example.com/" session-start="2003-09-
889 27T10:35:49">
890 - <vulnerability-probe id="probe-1234" time-stamp="23.124">
891 - <test-probe id="test-probe-1" test-script-ref="test-1">
892 - <http-probe>
893 - <request method="GET" connection="proxy.example.com:8080"
894 host="www.example.com:80" request-uri="/login.asp\" version="HTTP/1.1">
895 - <raw>
896   GET /login.asp\ HTTP/1.1
897   <eol />
898   Host: example.com:80
899   <eol />
900   User-Agent: SensePostData
901   <eol />
902   Content-Type: application/x-www-form-urlencoded
903   <eol />
904   Translate: f
905   <eol />
906 </raw>
907 - <parsed>
908   <header name="User-Agent" value="SensePostData" />
909   <header name="Content-Type" value="application/x-www-form-urlencoded" />
910 </parsed>
911 </request>
912 - <response>
913 - <raw>
914   HTTP/1.1 200 OK
915   <eol />
916   Server: Microsoft-IIS/5.0
917   <eol />
918   Date: Fri, 01 Aug 2003 02:28:12 GMT
919   <eol />
920   X-Powered-By: ASP.NET
921   <eol />
922   Connection: Keep-Alive
923   <eol />
924   Content-Length: 167
925   <eol />
926   Content-Type: text/html
927   <eol />
928 </raw>
929 - <parsed>
930   <statusline name="status-code" value="200" />
931   <header name="Server" value="Microsoft-IIS/5.0" />
932   <header name="Date" value="Fri, 01 Aug 2003 02:28:12 GMT" />

```

```

933 <header name="X-Powered-By" value="ASP.NET" />
934 <header name="Connection" value="Keep-Alive" />
935 <header name="Content-Length" value="167" />
936 <header name="Content-Type" value="text/html" />
937 </parsed>
938 </response>
939 </http-probe>
940 </test-probe>
941 - <vulnerability-description title="IIS Translate:f Source Code Disclosure"
942 version="2003-09-27JS-1" id="vulnerability-1">
943 - <!--
944 ===== Common to all blocks =====
945 - <!--
946 === summary ===
947 <summary>A vulnerability in IIS allows remote attackers to view the
948 source of offered server side scripts supported by IIS (such as ASP,
949 ASA, HTR, etc.) by using malformed "Translate: f" header.</summary>
950 - <!--
951 === description ===
952 - <description xml:lang="en">
953 A vulnerability in IIS allows remote attackers to view the source of
954 offered server side scripts supported by IIS (such as ASP, ASA, HTR,
955 etc.).
956 <xhtml:p />
957 This vulnerability is very dangerous since a lot of sensitive information
958 is kept in these files, as programmers often rely on the fact that the
959 source code is hidden from the user. The vulnerability involves sending
960 a special header with 'Translate: f' at the end of it, and then a trailing
961 back-slash '\ ' appended to the end of the URL. It cannot be exploited by
962 the standard browsers, but an exploit code below enables to test for this
963 problem.
964 <xhtml:p />
965 WebDAV implemented in Windows 2000 and Office 2000 (including
966 FrontPage 2000 and FrontPage 2000 Server extensions) is the source of
967 Translate:f problem.
968 <xhtml:p />
969 When someone makes request for ASP/ASA (or any other scriptable
970 page) and adds "Translate: f" into headers of HTTP GET request
971 (headers are not part of the URL, they are part of the raw HTTP
972 request), there is a serious security bug in Windows 2000 (when
973 unpatched by SP1) that in return gives complete ASP/ASA code instead
974 of processed file. It's necessary to add a trailing back-slash "\" to end
975 of requested URL to make this work.
976 <xhtml:p />

```

977 **"Translate:f" is legitimate header for WebDAV, it is used as it should**
978 **be - adding this to HTTP GET is a signal for the WebDAV component to**
979 **return the source code of the requested file and bypass processing. It is**
980 **used in FrontPage2000 and any WebDAV compatible client to get a file**
981 **for editing. It has to be accompanied by some other information, which**
982 **should prevent unauthorized users from viewing the source.**
983 **Unfortunately, a coding problem makes it possible to retrieve those files**
984 **by simply adding "Translate:f" in the header, and placing "\" at end of**
985 **request to the HTTP GET.**

986 `<xhtml:p />`

987 **It is a Windows 2000 bug, but because of FrontPage Server Extensions**
988 **2000 can be installed even on IIS 4.0 sites, it also affectes IIS 4.0. Many**
989 **IIS 4.0 sites will exhibit the "Translate: f" bug when web files are stored**
990 **on a shared (network) directory, this vulnerability has been fixed in the**
991 **past (see our previous article: Patch Available for the Virtualized UNC**
992 **Share Vulnerability).**

993 `</description>`

994 `- <!--`

995 `=== classification ===`

996 `<classification xmlns:wasm="urn:oasis:names:tc:wasm:1.0:..." name="was:severity"`
997 `value="75" />`

998 `- <!--`

999 `=== datum ===`

1000 `<datum name="avdln:date-found" type="date" value="2000-06-05" />`

1001 `<datum name="avdln:found-by" type="string" value="SecurITeam" />`

1002 `- <!--`

1003 `=== history ===`

1004 `<history uri="http://www.oasis-`

1005 `open.org/apps/org/workgroup/avdl/download.php/2927/avdlstruct1.xml" />`

1006 `- <!--`

1007 `===== vulnerability specific =====`

1008 `- <!--`

1009 `=== target ===`

1010 `- <target id="target-win2k">`

1011 `<os name="Microsoft Windows 2000 Version 2.1 Service Pack 3" />`

1012 `<arch name="Intel Pentium III" />`

1013 `<webserver name="IIS 4.0" />`

1014 `<webserver name="WebDAV" />`

1015 `<protocol name="HTTP 1.0" />`

1016 `</target>`

1017 `- <!--`

1018 `=== test ===`

1019 `- <test-script id="test-1" target-ref="target-win2k">`

1020 `- <!--`

1021 `Test description`

1022 `<declare name="proxy-host" type="host" />`

1023 `<declare name="proxy-port" type="integer" default="8080" />`

1024 `<declare name="host" type="host" />`

1025 `<declare name="port" type="integer" default="80" />`

1026 `<declare name="path" type="string" />`

1027 `<declare name="protocol" type="string" default="HTTP/1.1" />`

1028 `- <sequence repeat="1">`

```

1029 - <http-transaction>
1030 - <request>
1031   GET
1032   <space />
1033   /
1034   <var name="path" />
1035   <attack comment="GET ending in backslash">\</attack>
1036   <space />
1037   <var name="protocol" />
1038   <eol />
1039   Host:
1040   <space />
1041   <var name="host" />
1042   :
1043   <var name="port" />
1044   <eol />
1045   User-Agent:
1046   <space />
1047   SensePostData
1048   <eol />
1049   Content-Type:
1050   <space />
1051   application/x-www-form-urlencoded
1052   <eol />
1053   <attack comment="Should have required additional info to return
1054 source">Translate: f</attack>
1055   <eol />
1056   </request>
1057 - <response>
1058 - <expect status-code="200" reason-phrase="OK">
1059   <match-header name="Content-Type" value="application/octet-stream" />
1060   </expect>
1061   </response>
1062   </http-transaction>
1063   </sequence>
1064   </test-script>
1065   <remediation remedy-id="remedy-1" remedy-description="Apply the appropriate
1066 service pack for Windows XP" remedy-moddate="2003-09-27T10:35:49" remedy-
1067 vendor="Citadel" remedy-language="VBScript" remedy-
1068 href="http://vendor.remediation.com/library/q25688.vb" remedy-code="REM Copyright
1069 2003, Citadel Security Software, Inc. All Rights Reserved. All product names are
1070 trademarks or registered trademarks of their respective owners. Specifications
1071 subject to change without notice. REM Script Generated Automatically by skey at
1072 9/10/2003 2:04:30 PM Option Explicit HercClient.SetScriptReturnCode( 5 ) REM Failure
1073 Dim sVersion, sFull, sSP, bPassed bPassed = true If bPassed = true Then If
1074 HercClient.IsWindowsXP() = True then If HercClient.WindowsCSDVersion < Service Pack 1
1075 Then bPassed = True Else bPassed = False End If End If End If" />
1076 - <remediation remedy-id="remedy-2" remedy-description="For RPCSS, Apply the
1077 MS03-039 patch and the appropriate service pack for Windows XP" remedy-
1078 moddate="2003-09-27T10:35:49">

```



```
1079 <patch patch-vendorname="Microsoft" patch-
1080 href="http://download.microsoft.com/download/win2000platform/Patch/Q256888/NT5/EN-
1081 US/Q256888_W2K_SP1_x86_en.EXE" patch-ref="Q256888_W2K_SP1_x86_en" patch-
1082 lang="english" patch-switches="/q /z" patch-method="install" />
1083 <patch patch-vendorname="Microsoft" patch-
1084 href="http://download.microsoft.com/download/c/d/d/cdd7ac92-e4cc-4b1e-bc2f-
1085 7a61b46b23bf/WindowsXP-KB824146-x86-ENU.exe" patch-ref="WindowsXP-KB824146-x86-
1086 ENU.exe" patch-lang="english" patch-switches="/q /z" patch-method="install" />
1087 </remediation>
1088 </vulnerability-description>
1089 </vulnerability-probe>
1090 </session>
1091 </avdl>
```

1092

Appendix B. Acknowledgments

1093 The AVDL Technical Committee would like to acknowledge earlier efforts in promotion of
1094 application vulnerabilities and standardization of their representation and interchange. Their work
1095 inspired many ideas incorporated into the AVDL standard.

1096 Open Vulnerability Assessment Language developed at the Mitre Corporation “is the common
1097 language for security experts to discuss and agree upon technical details about how to check for
1098 the presence of a vulnerability on a computer system”. Using SQL, OVAL queries are based on
1099 broadly recognized Common Vulnerabilities and Exposures (CVE) database and by “specifying
1100 logical conditions on the values of system characteristics and configuration attributes, OVAL
1101 queries characterize exactly which systems are susceptible to a given vulnerability.”

1102 VulnXML developed by a OWASP team led by Mark Curphey “could be used by automated
1103 assessment tools to test for known security issues”. Closely related and also developed at
1104 OWASP was Application Security Attack Components or ASAC which “is a basic classification
1105 scheme of web application security issues. The aim of this project was to create a common
1106 language and a consensus understanding among the industry to describe the same issue in the
1107 same way.” Their work continues at OASIS Web Application Security TC.

1108

1109

Appendix C. 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

1110

1111

Appendix D. Notices

1112 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
1113 that might be claimed to pertain to the implementation or use of the technology described in this
1114 document or the extent to which any license under such rights might or might not be available;
1115 neither does it represent that it has made any effort to identify any such rights. Information on
1116 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
1117 website. Copies of claims of rights made available for publication and any assurances of licenses
1118 to be made available, or the result of an attempt made to obtain a general license or permission
1119 for the use of such proprietary rights by implementers or users of this specification, can be
1120 obtained from the OASIS Executive Director.

1121

1122 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
1123 applications, or other proprietary rights that may cover technology that may be required to
1124 implement this specification. Please address the information to the OASIS Executive Director.

1125

1126 Copyright © OASIS Open 2002. *All Rights Reserved.*

1127

1128 This document and translations of it may be copied and furnished to others, and derivative works
1129 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
1130 published and distributed, in whole or in part, without restriction of any kind, provided that the
1131 above copyright notice and this paragraph are included on all such copies and derivative works.
1132 However, this document itself does not be modified in any way, such as by removing the
1133 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS
1134 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
1135 Property Rights document must be followed, or as required to translate it into languages other
1136 than English.

1137

1138 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
1139 successors or assigns.

1140

1141 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1142 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
1143 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE
1144 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1145 PARTICULAR PURPOSE.