# Application Vulnerability Description Language

## Committee Draft Version 1.0, 15 March 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 standard for creating a uniform method of describing application security vulnerabilities using XML.

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

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

**Status:**

This document is the AVDL Technical Committee Draft. Please send comments to the editors.

Committee members should send comments on this specification to avdl@lists.oasis-open.org. Others should subscribe to and send comments to avdl-comment@lists.oasis-open.org. To subscribe, send an email message to avdl-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

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

**Eratta:**

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

# Table of Contents

# Introduction

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

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

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

Vulnerability information may include:

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

AVDL is capable of describing either type of information.

## 1.1 Notations and Terminology

### 1.1.1 Notations

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

### 1.1.2 Terminology

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

- **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.
- **Vulnerability Description Title** – This descriptor defines the vulnerability within the Test Probe.
- **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).

## 1.2 Requirements

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

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

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

## 1.3 Out of Scope

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

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

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

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

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

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

# 2 AVDL Output

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

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

## 2.1 AVDL File Root

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

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

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

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

## 2.2 Traversal

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

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

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

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

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

### 2.2.1 Traversal Container

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

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

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

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

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

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

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

```
        <raw>GET / HTTP/1.0 Connection: Close Host: 172.16.50.31
        User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0) Pragma: no-
        cache</raw>
        <parsed>
            <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="Pragma" value="no-cache"/>
            <query value=""/>
            <content value=""/>
        </parsed>
        </request>
        <response>
            <raw>HTTP/1.1 302 Object moved Server: Microsoft-IIS/5.0 Date: Tue, 10 Feb
            2004 13:29:39 GMT Location: banklogin.asp?serviceName=
            FreebankCaastAccess&amp;templateName=prod_sel.forte&amp;source=
            Freebank&amp;AD_REFERRING_URL= http://www.Freebank.com Connection:
            Keep-Alive Content-Length: 251 Content-Type: text/html Cache-control: private
            Set-Cookie: ASPSESSIONIDGGQQQUIU= GJABGOGAEBIONOCNAGGKNLNF;
            path=/&lt;head&gt;&lt;title&gt;Object moved&lt;/title&gt;&lt;/head&gt;&lt;
            body&gt;&lt;h1&gt;Object Moved&lt;/h1&gt;This object may be found &lt;a
            HREF="banklogin.asp?serviceName=FreebankCaastAccess&amp;
            templateName=prod_sel.forte&amp;source=Freebank&amp;
            AD_REFERRING_URL= http://www.Freebank.com"&gt;here&lt;
            /a&gt;.&lt;/body&gt;</raw>
        <parsed>
            <statusline value="HTTP/1.1 302 Object moved"/>
            <header name="Server" value="Microsoft-IIS/5.0"/>
            <header name="Date" value="Tue, 10 Feb 2004 13:29:39 GMT"/>
            <header name="Location" value="banklogin.asp?serviceName=
            FreebankCaastAccess&amp;templateName=prod_sel.forte&amp;source=
            Freebank&amp;AD_REFERRING_URL=http://www.Freebank.com"/>
            <header name="Connection" value="Keep-Alive"/>
            <header name="Content-Length" value="251"/>
            <header name="Content-Type" value="text/html"/>
            <header name="Cache-control" value="private"/>
            <content>
                <href uri="banklogin.asp?serviceName=FreebankCaastAccess&amp;
                templateName=prod_sel.forte&amp;source=
                Freebank&amp;AD_REFERRING_URL=http://www.Freebank.com"
                type="static" persistence="export"/>
                <href uri="banklogin.asp?serviceName=FreebankCaastAccess&amp;
                templateName=prod_sel.forte&amp;source=
                Freebank&amp;AD_REFERRING_URL=http://www.Freebank.com"
                type="static" persistence="export"/>
                <href uri="banklogin.asp" type="static" persistence="export"/>
            </content>
        </parsed>
    </response>
</http-traversal>
```

## 2.3 Vulnerability Probe

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

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

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

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

### 2.3.1 Vulnerability Probe Container

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

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

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

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

```
<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:
            ASPSESSIONIDGGQQQUIU=GJABGOGAEBIONOCNAGGKNLNF;
            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>
```

### 2.3.2 Vulnerability Properties

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

- Summary - a brief description of the vulnerability
- Description - a detailed description of the vulnerability
- Classification - a unique identifier for the vulnerability
- Datum - metadata about the vulnerability
- History - the version of the vulnerability that was used

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

Subsequent sections will provide more detail to the Vulnerability properties.

#### 2.3.2.1 Summary

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

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

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

#### 2.3.2.2 Description

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

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

### 2.3.2.3 Classification

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

## 2.3.3 Vulnerability Specific

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

### 2.3.3.1 Test

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

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

### 2.3.3.2 Test description

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

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

```
<declare name="path" type="string"/>
<declare name="protocol" type="string" default="HTTP/1.1"/>
<declare name="host" type="host"/>
<declare name="port" type="integer" default="80"/>
<sequence>
    <http-transaction>
        <request>GET &lt;var name="path"/&gt; &lt;var name="protocol"/&gt; Referer:
        http://&lt;var name="host"/&gt;:&lt;var name="port"/&gt;/
        Connection: Close
        Host: &lt;var name="host"/&gt;
        User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
        Pragma: no-cache
        Translate: f
        Cookie: ASPSESSIONIDGGQQQUIU=GJABGOGAEBIONOCNAGGKNLNF;
        CustomCookie=WebInspect</request>
        <response>
```

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

### 2.3.3.3 Remediation

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

```
<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>
```

# Appendix A. Acknowledgments

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

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

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

# 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, |

# Appendix C. Notices

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

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