**SANS GSEC PRACTICAL**

# Vulnerability naming schemes and description languages: CVE, Bugtraq, AVDL and VulnXML

**Michael Rohse**

**GSEC Practical Version 1.4b (1)**
**April 22, 2003**

# Vulnerability naming schemes and description languages: CVE, Bugtraq, AVDL and VulnXML

## Abstract:

Administrators and security experts are usually flooded with attack and vulnerability information, generated by the different security products like Firewalls, Intrusion Detection systems and Vulnerability Assessment tools. To react in a timely manner it is essential that these tools are naming events in a common way. The today's de facto standards CVE (Common Vulnerability Exposures) and the SecurityFocus Bugtraq database will be presented.
CVE and Bugtraq have some limitations, they do not describe the vulnerability in enough detail and a common format.
These limitations inspired two new proposals: AVDL (Application Vulnerability Description Language) and VulnXML. With them it will be possible to directly import a describing XML document into a scanning tool and the tool will generate and launch the vulnerability scan. AVDL and VulnXML will be described and discussed in this paper.

**Table of content:**

## I. Introduction:

A major problem in our networked society is the growing number of security issues. The CERT (Computer Emergency Response Team) is regularly publishing statistics, which show an enormous rise in vulnerabilities especially in the last two years [1]: It went up from 2437 vulnerabilities in the year 2001 to 4129 in 2002! Companies need to protect their assets against these threats with the right combination of security products:

- Firewalls are used to control the data flow between the internal network and the Internet.
- Antivirus software is used to analyze and filter out suspicious code.
- Intrusion Detection systems are constantly monitoring the local machines and the network traffic.
- Vulnerability Assessment tools are scanning each system on a regular basis against potential weaknesses.
- Public Key Infrastructure (PKI) and Directory Services are used to organize and authenticate users.

All the above tools are necessary for defense in depth, but it is also essential to have good security policies. These policies must define the allowed usage of the whole IT infrastructure and therefore the configuration of all security products can be derived from these policies.

Unfortunately most of the products are generating output in their own format, especially if they are from different vendors. Similar events, detected by different products are difficult to correlate, especially if the events are named differently. To address this problem, an initiative called “common vulnerability enumeration” was started by MITRE in 1999. Today CVE stands for “Common Vulnerabilities and Exposures” and is the de facto standard in the security industry. CVE has some limitations, which resulted in the proposal of two vulnerability description languages, called VulnXML and ADVL. These languages could be used to describe vulnerability in the extended markup language XML. Data in this format could then be used directly by scanning tools, which could import it and generate some scans out of it. It also would make the job of event correlation easier, because the different tools could generate their event logs in a similar XML format. Data in the same format is easier to interpret by log analysis and event correlation tools.

## II. Common Vulnerabilities and Exposures (CVE):

The idea of CVE was first introduced in the paper “Towards a Common Enumeration of Vulnerabilities” [2] written by David E. Mann, Steven M. Christey in January 1999. David and Steven had the problem, that they used different security tools within their corporation. When these tools generated similar events, they had to determine as soon as possible, if these events resulted from the same vulnerability or from different ones. So they decided to build a central database that can be used as a reference to the vulnerability information from different source. This reference database had to solve the following problems:

1) **Inconsistent naming conventions**: David and Steve quoted an example of a Network File System (NFS) vulnerability which was named "nfs-guess" by X-Force, "NFS file handle guessing check" by Cybercop and "SunOS NFS Jumbo and fsirand Patches" in a CERT Advisory. This made it nearly impossible to correlate event information between the different tools.

2) **Different perspectives** of the same vulnerability detected by different tools: A virus scanner will search for an attack pattern, while a vulnerability assessment tool will search for an installed application. David and Steven used two databases, one for the events of their intrusion detection tools and one for the events of their vulnerability assessment tools. Having one reference database would improve their ability to correlate between the two databases and help them to find identical weaknesses.

3) **Free and completed distribution**: Some of the vulnerability information they used was copyrighted, most of the sources were not complete and some were publishing only the widespread and dangerous items. They wanted to be able to publish a complete set of vulnerability information in the form of a freely distributable database.

One additional benefit of having one reference database would be the possibility to compare different tools against each other. Vendors claim to be able to identify a large number of vulnerabilities. If these numbers could be verified against a common basis, they are really comparable. This would help them to decide which tool fit the best for a certain environment.

Their initiative led to the foundation of CVE, a freely distributable database, which is now widely used. On April 3, 2003 it contained 2573 entries and 3109 candidates. A candidate is a "work in progress" entry.

Today CVE is the de facto standard in the industry, a large number (122) of security products are referencing [3] its information and 79 Organizations participate. The listed products are "CVE-compatible", which means that the output contains CVE references that are up-to-date and searchable. CVE is organized and hosted by MITRE [4], a not-for-profit corporation funded by the DOD, FAA and IRS.
CVE also defines a naming process [5] starting with the detection of a new vulnerability and ending with the admission into the list.

**The different phases of the CVE naming process:**

1) **Discovery** – The potential exposure is discovered.

2) **Public Announcement** – This info is announced in a public forum like a newsgroup, a mailing list or a security advisory. After the announcement, the information is submitted to the CVE Candidate Numbering Authority (CNA).

3) **Assignment** – the Candidate Numbering Authority verifies, if the new candidate is already known and if not assigns it an unique identifier, called candidate (CAN) in the following format
"CAN-< four digit year>-<four digit number>"

4) **Proposal** – the candidate is handed over to the Editorial Board. The Editorial Board consists of well known and respected representatives from major corporations like Cisco, Symantec, SUN, Microsoft, research / educational institutions like SANS, Carnegie Mellon University and governmental organisations like NIST and NSA. The full member list [6] is available on their website.
The members discuss the candidate and vote for it. Possible votes are:

- Accept
- Reject
- Recast: major changes are necessary
- Modify: minor changes are necessary
- No opinion or
- Still working on it.

5) **Discussion** - When a "Modify" or "Recast" is voted, the candidate needs to be discussed again. It is sent back again for voting.

6) **Decision** – The first step is an interim decision, where all members are asked to vote with either an "Accept" or "Reject". If votes are given with attached detailed comments, the candidate is sent back to the modification phase. If the candidate is finally accepted or rejected, all members are informed.

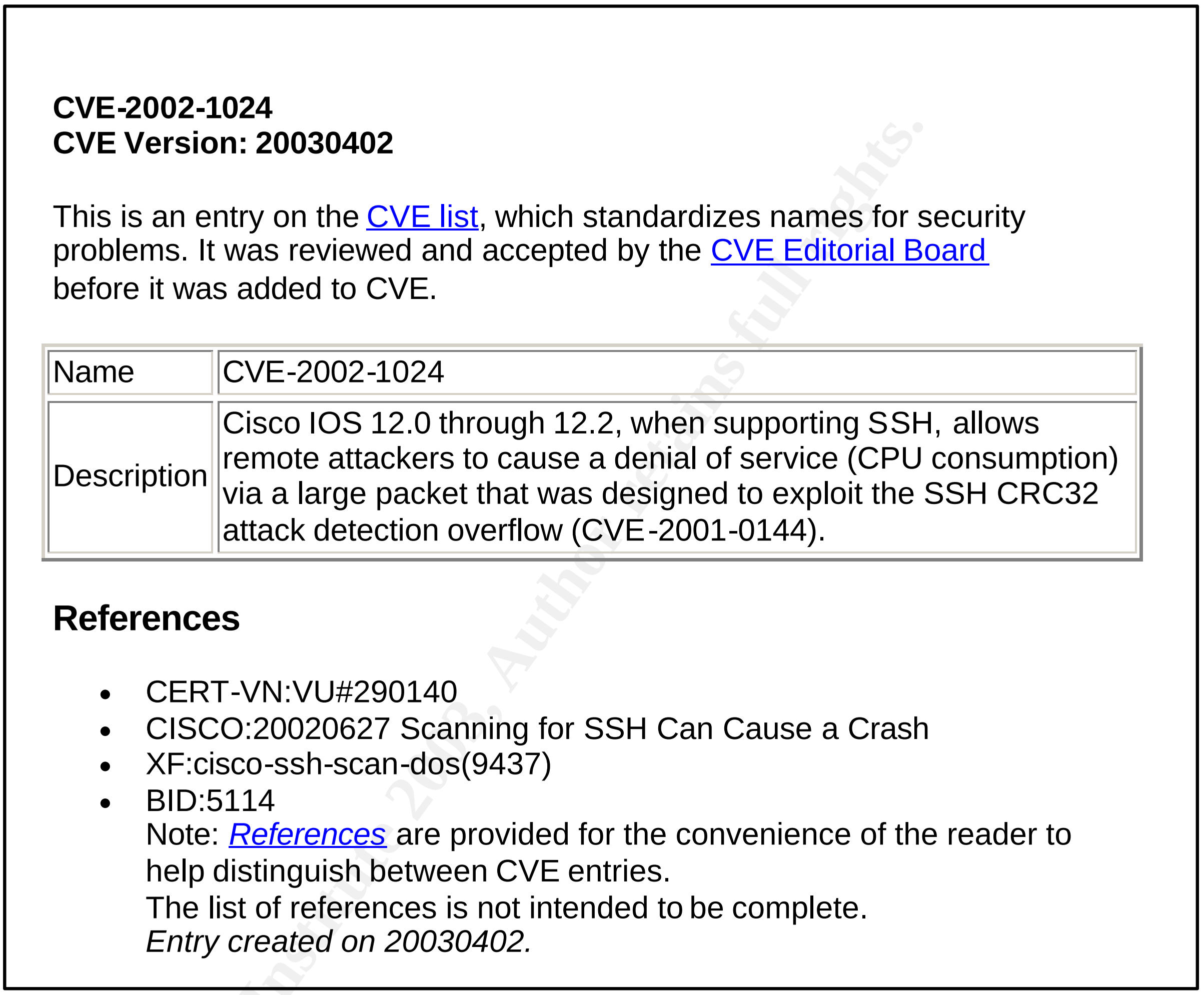
7) **Publication** – The candidate is renamed from CAN-year-xxxx to CVE-year-xxxx, posted on the website and a new version of the whole CVE database/list is created. The "year-xxxx" identifier is kept when the rename process is done.

8) **Reassessment** – Eventually an entry has to be modified after publication. This could happen if the exposure has a wider impact. The reassessment has to follow the same steps as a new candidate.

9) **Depreciation** – Eventually the board decides to set the status of an entry to inactive or to split up an entry into lower level entries.

A final CVE entry consists of a descriptive name, a brief description and references to the vulnerability.

The next picture shows an example of a CVE Entry:



**CVE-2002-1024**
**CVE Version: 20030402**

This is an entry on the CVE list, which standardizes names for security problems. It was reviewed and accepted by the CVE Editorial Board before it was added to CVE.

| | |
|---|---|
| Name | CVE-2002-1024 |
| Description | Cisco IOS 12.0 through 12.2, when supporting SSH, allows remote attackers to cause a denial of service (CPU consumption) via a large packet that was designed to exploit the SSH CRC32 attack detection overflow (CVE-2001-0144). |

## References

- CERT-VN:VU#290140
- CISCO:20020627 Scanning for SSH Can Cause a Crash
- XF:cisco-ssh-scan-dos(9437)
- BID:5114
  Note: *References* are provided for the convenience of the reader to help distinguish between CVE entries.
  The list of references is not intended to be complete.
  *Entry created on 20030402.*

The References are pointing to CERT, CISCO, Internet Security Systems X-Force and the Bugtraq ID.

## III. Bugtraq

Another valuable source of information is the Bugtraq [7] mailing list, hosted and organized by the vendor SecurityFocus. Due to the high traffic and to make sure a certain level of discussion culture is maintained, a person moderates it. Its mission is to discuss and announce security vulnerabilities in depth. Security hardware and software should resist all possible attacks, even some that use knowledge about the internal architecture of a product.

This mailing list follows the “full disclosure” philosophy, which means, that exploits should be presented in detail, so everyone can verify and test it.
The advantage of this philosophy is:

- Vendors have to provide security fixes fast, because the vulnerability is well described and it is usually easy to exploit.
- A lot of people can verify the exploit and use it to break similar systems and/or programs
- People can learn from other people’s errors.

The other side of the coin is, that there are not only good guys in the world and it is quite possible that these people have also joined the mailing list.

There is a recommendation that a newly discovered exploit should first be mailed to the affected vendor and given approximately one week to respond.
If the vendor does not react, the exploit is posted to the mailing list.
If the vendor reacts, it is up to the author of the exploit to decide, how much time the vendor has to fix the problem.
Certainly the impact of the new vulnerability should also be calculated into the amount of time given.

The content and correctness of the different postings is only verified by SecurityFocus, the company that is providing and moderating the list. Of course the normal list discussion process helps to correct any initial errors. There is no vendor neutral forum / editorial board as it exists at CVE.
SecurityFocus is hosting also some other mailing lists which are for example covering incidents, penetration test, forensics, just to name a few. A searchable database [8] of the vulnerabilities is available, which consists of over 7300 entries as of April 2003. Searchable items are keyword, title, vendor, Bugtraq id or corresponding CVE entry.

Bugtraq entries are classified into ten categories; a description of these categories is available at the help page of each Bugtraq id[9]:

- Boundary Condition Error,
- Access Validation Error,
- Input Validation Error
- Origin Validation Error
- Failure to Handle Exceptional Conditions

- Race Condition Errors
- Serialization Errors
- Atomicity Errors
- Environment Errors
- Configuration Errors

Then it is described, whether it is exploitable locally or remote, if it is published, updated and which products are vulnerable or not vulnerable.

The entries in the Bugtraq database are presented in form of 5 sub pages. As an example the Bugtraq id 6070 [10]is shown here, a Microsoft IIS WebDAV Denial of Service Vulnerability (it is stripped to make it better readable):

Page 1 Info:

| | |
|---|---|
| bugtraq id | 6070 |
| object | |
| class | Failure to Handle Exceptional Conditions |
| cve | CAN-2002-1182 |
| remote | Yes |
| local | No |
| published | Oct 31, 2002 |
| updated | Oct 31, 2002 |
| vulnerable | Microsoft IIS 5.0<br>+ Microsoft Windows 2000 Advanced Server<br>- Microsoft Windows 2000 Advanced Server SP1<br>- Microsoft Windows 2000 Advanced Server SP2<br><some items deleted> |
| not vulnerable | |

Page 2 Discussion:

A denial of service vulnerability has been reported for Microsoft IIS 5 and 5.1. This vulnerability is related to how IIS allocates memory for WebDAV requests. Any specially crafted WebDAV requests may result in IIS allocating an extremely large amount of memory on the server. Several malformed requests sent to the server will result in the vulnerable system failing to respond to further legitimate requests for service. This vulnerability affects IIS 5.0 and 5.1 only.

This vulnerability was originally described in Bugtraq ID 6068. It is now being assigned its own Bugtraq ID.

### Page 3 Exploit:

Currently we are not aware of any exploits for this issue. If you feel we are in error or are aware of more recent information, please mail us at: vuldb@securityfocus.com <mailto:vuldb@securityfocus.com>.

### Page 4 Solution:

**Workaround:**
The use of the IIS Lockdown Tool provided by Microsoft may help prevent exploitation of this vulnerability.
**Solution:**
Patches are available:
Microsoft IIS 5.0:
Microsoft Patch Q327696: Internet Information Services Security Roll-up Package
http://www.microsoft.com/windows2000/downloads/security/q327696/….
<some items deleted and the url shortened>

### Page 5 Credit:

Credit:
Discovery of this vulnerability credited to Mark Litchfield of Next Generation Security Software Ltd. (http://www.nextgenss.com).

References:

| | |
|---|---|
| Message: | Microsoft Internet Information Server 5/5.1 Denial of Service (#NISR31102002) |
| Web page: | Microsoft Security Bulletin MS02-062<br>(Microsoft) |

Both CVE and Bugtraq are good and have set standards in naming the vulnerabilities. The level of detail, they are presenting vulnerability with, is not sufficient enough, so that it can be used directly as a pattern for scanners. This issue is addressed by the following two proposals VulnXML and AVDL.

## IV. VulnXML

Security researchers are publishing their findings in a way, which is either specific to a certain tool or is in free-form textual format. This makes it difficult to share the original information between different tools, either it is incompatible or has to be created from scratch. VulnXML should address this problem.

VulnXML [11] is a Web Application Security Vulnerability Description Language written in the extended mark-up language XML format. It is an open standard and belongs to the Open Web Application Security Project [12] (OWASP).
OWASP is an open source community with the mission to help people building web applications with a better level of security.
The project leaders for VulnXML are Rogan Dawes, Mark Curphey, Yuval Ben-Itzak and Jennifer Thrap. The project was started in 2002.
When this paper was written (April 2003) VulnXML was not finished, only the initial Vision Document, the VulnXML DTD Version 1.1 and one VulnXML example was available on their project web page.

VulnXML should be used to describe application level vulnerabilities, it is not intended to cover network based ones. These kinds of attacks are right now well covered with the open source network based Intrusion detection system SNORT and its bundled rule set.

The following listing [13] describes an IIS Chunked Encoded Buffer Overflow in VulnXML. You can see the references to Bugtraq, CVE, Microsoft and CERT at the beginning, followed by a description of the targeted system and finally the test of the vulnerability:

```
 <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
  <!DOCTYPE WebApplicationTest (View Source for full doctype...)>
- <WebApplicationTest>
  - <TestDescription>
      <TestName>OWASP-00002</TestName>
      <TestVersion>0.0</TestVersion>
      <DateReleased>2002-04-10</DateReleased>
      <DateUpdated>2002-04-30</DateUpdated>
      <OWASP_Class class="Overflows"
        URL="http://www.owasp.org/asac/" />
    - <References>
        <Reference database="Bugtraq"
          URL="http://www.securityfocus.com/bid/4485" />
        <Reference database="CVE"
          URL="http://cve.mitre.org/cgi-
          bin/cvename.cgi?name=CAN-2002-0079" />
        <Reference database="Microsoft"
          URL="http://www.microsoft.com/technet/security/b
          ulletin/ms02-018.asp" />
        <Reference database="Cert"
          URL="http://www.cert.org/advisories/CA-2002-
          09.html" />
      </References>
      <Copyright>Public Domain</Copyright>
      <TestProtocol protocol="HTTP" />
      <MayProxy value="True" />
      <Description>Buffer overflow in the chunked encoding
        transfer mechanism in Internet Information Server (IIS)
        4.0 and 5.0 Active Server Pages allows attackers to cause
        a denial of service or execute arbitrary code.</Description>
    - <ApplicableTo>
      - <Platform>
          <OS>Windows</OS>
          <Arch>i386</Arch>
        </Platform>
        <WebServer>Microsoft-IIS</WebServer>
        <ApplicationServer />
      </ApplicableTo>
      <Affects scope="server" />
    - <TriggerOn event="file">
        <Match type="regex">.*.asp</Match>
      </TriggerOn>
      <Impact>The attacker can cause the web server to crash and
        restart, and could potentially execute arbitrary code on
        the web server</Impact>
      <Severity value="high" />
      <Recommendation>Delete sample ASP scripts to deter bulk
        scanners. Install the patch supplied by Microsoft as soon
        as it is available.</Recommendation>
      <AlertOn result="SUCCESS" />
    </TestDescription>
    <Inputs />
  - <Connection>
    - <Step name="step1">
      - <Request>
        - <MessageHeader>
            <Method encoding="text">POST</Method>
            <URI
              encoding="text">${scheme}://${host}:${por
              t}/${path}/${file}</URI>
            <Version encoding="text">HTTP/1.1</Version>
            <Header name="Accept"
              encoding="text">*/*</Header>
            <Header name="Host"
              encoding="text">${host}</Header>
            <Header name="Transfer-Encoding"
              encoding="text">chunked</Header>
          </MessageHeader>
        - <MessageBody>
```

```
        <Content-Type encoding="text">application/x-
          www-form-urlencoded</Content-Type>
        <Content-Length length="auto" />
        <Separator encoding="text" />
        <Item
          encoding="base64">MQpFCjAKCgoK</Item>
      </MessageBody>
    </Request>
  - <Response>
    - <SetVariable name="ResponseCode" type="string">
        <Description>HTTP Response code</Description>
        <Source source="status-
          line">^.*\s(\d\d\d)\s</Source>
      </SetVariable>
    - <SetVariable name="redir302" type="string">
        <Description>See if we got a custom 404
          handler, correctly implemented using a
          redirection</Description>
        <Source source="message-header">Location:
          (.*)$</Source>
      </SetVariable>
    - <SetVariable name="body404" type="string">
        <Description>See if we got a custom 404
          handler, incorrectly implemented using a
          return code of 200</Description>
        <Source source="message-body">(404 Not
          Found)</Source>
      </SetVariable>
    - <SetVariable name="unpatched" type="string">
        <Description>An unpatched server returns
          "(0x80004005)<br>Unspecified</Description
          >
        <Source source="message-
          body">(\(0x80004005\)<br>Unspecified)</
          Source>
      </SetVariable>
    - <SetVariable name="patched" type="string">
        <Description>A patched server returns
          "(0x80004005)<br>Request</Description>
        <Source source="message-
          body">(\(0x80004005\)<br>Request)</Sour
          ce>
      </SetVariable>
    </Response>
  - <TestCriteria type="FAILURE">
      <ErrorMessage>The page was not
        found</ErrorMessage>
    - <Compare variable="${ResponseCode}" test="equals"
        value="200">
```

```
            <Compare variable="${body404}"
              test="notequals" value="" />
          </Compare>
          <Compare variable="${ResponseCode}" test="equals"
            value="404" />
          <Compare variable="${ResponseCode}" test="equals"
            value="302" />
          <Compare variable="${ResponseCode}" test="equals"
            value="500" />
        </TestCriteria>
      - <TestCriteria type="FAILURE">
          <Compare variable="${patched}" test="notequals"
            value="" />
        </TestCriteria>
      - <TestCriteria type="SUCCESS">
          <Compare variable="${unpatched}" test="notequals"
            value="" />
        </TestCriteria>
      </Step>
    </Connection>
  </WebApplicationTest>
```

Two tools will be able to read and import VulnXML documents, a commercial one from the vendor KAVADO called ScanDo[14] and a free one named WebScarab[15].

**WebScarab** is a web application scanner written in Java to be platform independent. WebScarab will be able to check for both static and dynamic vulnerabilities, but it is focused on dynamic ones. A static one for example is a URL that can be accessed without any authorization, a dynamic one is for example a faked cookie to change the prize of an item in a shopping kart of an Online-store. As VulnXML, WebScarab is at the time of writing this paper (April 2003) still a work in progress. It consists of different parts, which are covering the standard phases of an assessment:

1. Discovery
2. Analysis and Test Case Creation
3. Test Execution
4. Reporting

The components are:

Spider, Proxy, Analysis Engine, VulnXML Parser, Attack Engine, DataStore, DataStore API, GUI and Reporting.

All parts will be described in the following paragraphs; source of the information is the WebScarab Vision document [16]:

**Spider -** It accepts an URL (Uniform Resource Locator) and if necessary a username/password combination as a starting point and browses the entire website searching for applications, that can be tested. It will follow all links that it is aware of. The spider will inspect all http requests and responses and looks for cookies, referrer headers, links etc. It stores all acquired information in the DataStore. The spider should emulate human behaviour; this makes some kind of pre-processing necessary. Forms need to be processed in an intelligent way; no spider is able to generate all possible entries. An input field in a search engine like Google needs to be recognized and processed differently as a customer login field with an expected username / password combination at the webpage of the internet bookstore Amazon
As a base product for this part, an already existing spider called WebSPHINX [17] will be used.

**Proxy** – a transparent reverse proxy is an alternative form of accessing a website. The user configures her/his browser to use this proxy; the proxy forwards all requests and responses back to the user's browser. The proxy acts as an http recorder and sends all relevant and interesting information to the DataStore. The proxy is used to manually test a specific URL or a specific item, for example a certain web page with embedded form entries. A user can navigate much more intelligent in a complex website than a spider, therefore the proxy is an intelligent user controlled alternative to the spider. The user interaction is especially necessary for websites with a lot of dynamic content, which is generated at runtime. This kind of website is difficult to browse automatically, because its content is based on some previous input.
As a base for this part, an already existing proxy called Muffin [18] will be used.

**Analysis engine** – the heart of WebScarab. It consists of different modules for different attack types like buffer overflows and cross-site scriptings. It generates a set of test cases from the data it will be feeded with. The modular concept makes it easier to enhance the engine, because each single module can be improved independently.

**VulnXML parser** – sub-component of the analysis engine. The VulnXML parser translates the XML documents into test cases.

**Attack Engine** – the attack engine launches all attacks it gets from the analysis engine and is listening for successful responses.

**DataStore** – a DataStore, which collects data generated by the spider, and the proxy, configuration data, user session data and GUI settings. Most likely it will be a relational database like MySQL or Oracle, but other forms of storage are also possible, if there are any special requirements.

**DataStore API** – an application-programming interface, so access to the DataStore is independent from WebScarab. Different DataStores may be used depending on the requirements.

**GUI** – the graphical interface to interact with WebScarab. This GUI is needed to make configuration changes to the different components like the spider and the proxy. This is used to import modules, view the progress during scans and view and generate reports.

**Reporting** – reports will be initiated via the GUI and generated in XML format. The XML file will be presented with XSLT as XHTML.

All components can be installed on one machine or distributed to different machines. This could be helpful in an enterprise scenario, where the DataStore might be implemented as a highly available database cluster that will be fed from multiple instances of spiders and proxies concurrently.

Another approach to describing vulnerabilities is AVDL, which seems to be very similar to VulnXML.

## V. AVDL Application Vulnerability Description Language

The application vulnerability description language (AVDL) is a new interoperability standard written in the extended markup language XML. Five application security vendors are proposing AVDL:

1. CITADEL SECURITY SOFTWARE the maker of Hercules, a vulnerability remediation solution
2. GUARDEDNET has a security risk management software called neuSECURE
3. NETCONTINUUM builds a web security appliance called NC-1000
4. SPI DYNAMICS are programming a web application security assessment software called WebInspect
5. TEROS are building security appliances

Their tools will be able to process data written in AVDL. With a common format their tools can work together:

- Assessment software scans an application, detects vulnerability, generates AVDL output and sends this data to a security appliance that blocks an attack before it can reach the application.
- Reporting tools could correlate AVDL events from different products.

AVDL will be an interoperable protocol and is focusing on vulnerabilities, which happen at the application level layer of the Hypertext Transfer Protocol (HTTP) 1.0 and 1.1

When this paper was written (April 2003), AVDL has just been announced. It will be discussed and developed in a new OASIS [19] technical committee. OASIS is a consortium with the mission to develop e-business standards; it consists of over 600 corporate and individual members participating in the discussion process. The first candidate should be published in Q3 2003 and the first final revision of the AVDL 1.0 specification is targeted for Q4 2003.

The vendors are proposing this standard, because they have been asked by customers to build interoperable applications. The customers wanted to choose the best available solutions and don't want to be limited to one vendor because of potential missing interaction.

## VI. Discussion and Conclusion

CVE and Bugtraq are well known and respected in the industry. SecurityFocus, the company that is providing Bugtraq has been recently acquired by Symantec, which raised some concerns about the independence and objectivity of it. The last few months have shown that nothing has changed, even flaws in Symantec software are still posted [20].

Bugtraq is superior compared to CVE when the speed of the communication and the wealth of information are considered. A lot of experienced people are interacting within Bugtraq and all other mailing lists that SecurityFocus is providing.
The disadvantage is the missing vendor-neutral verification and selection process, which makes it a little difficult to decide at discovery time, whether a new exploit is real and dangerous. After some time and discussion the decision is easier, because people have verified and tested it.

CVE is supported by most of the security vendors and its common naming scheme makes the life of security professionals easier. The defined naming process guarantees that (hopefully) no false alarms are published. On the other hand, it takes some time until a candidate becomes an entry. Some security tools are only referencing entries, not candidates. Speeding up the process would be helpful both for vendors and their customers.

VulnXML is a promising approach. The possibilities of this language will be shown both with a commercial and a free tool. The competition between a vendor and the user community makes it very interesting to see, which party is able to get the most out of it. The printed example of the IIS vulnerability shows the potential of a description language. A standard alone is worthless, so perhaps

VulnXML in combination with WebScarab can be as successful as SNORT[21] is in the network based Intrusion Detection area.
The success of WebScarab depends on the usability and timely availability of the pattern database. The user community (or in case of KAVADO a small vendor) is responsible for that. With SNORT a user community has shown that it can very well compete against the vendors.
A lot of network-based vulnerabilities are described with an attached SNORT rule. This rule can be incorporated into tools used to search for attacks that exploit this vulnerability. Most network based intrusion detection systems are able to import SNORT rules or are completely compatible, which means that they have rewritten their signatures into SNORT syntax. SNORT had set a standard in its area.

It is not completely clear, how much AVDL differs to VulnXML; but it's probably the interoperability and the approach to cover a broader product range. AVDL was introduced approximately one year later than VulnXML. In a mailing list posting one person raised some concerns that vendors try to earn the fruits of the pre-work done by the public community. The first implementation of the interoperability functions will be proprietary, because there is no standard yet. They claim to make future revisions public domain. One possible weakness could be that only a handful small vendors are cooperating and that no security "gorilla" has joined as of April 2003. Customers probably don't want standards proposed without the support of large companies.

VulnXML and AVDL are limited to Web based applications. A lot of applications are developed with or ported to this kind of interface and are also accessible from the Internet; that makes them vulnerable. Application level scans are complementary to network- and host-based scans. To make sure, that the business is protected, all applications should be scanned with the same intensity and regularity as it is done in the intrusion detection area with hosts and networks.

To achieve interoperability between different intrusion detection systems, the Internet Engineering Task Force (IETF) is actually (as of April 2003) discussing two drafts. The first one describes the "Intrusion Detection Exchange Protocol (IDXP)" [22], the second one the "Intrusion Detection Message Exchange Requirements"[23]. These drafts could probably be extended to some kind of universal interoperability in the security area and should be considered by the VulnXML and AVDL teams.

All in all the future is promising. Some new tools and standards are visible on the horizon and a lot of effort is under the way to achieve interoperability, which will help us all.

---

[1] CERT/CC Statistics 1988-2002
http://www.cert.org/stats/
Page last updated: 16-Apr-2003

[2] "Towards a Common Enumeration of Vulnerabilities"
David E. Mann and Steven M. Christey
9-Jan-1999
http://cve.mitre.org/docs/cerias.html
Page accessed: 22-Apr-2003

[3] CVE-Compatible Products and Services
http://www.cve.mitre.org/compatible/
Page last updated: 9-Apr-2003

[4] MITRE Homepage
http://www.mitre.org/
Page last updated: 27-Mar-2003

[5] The CVE Naming Process
http://cve.mitre.org/docs/docs2000/naming_process.html
Page last updated: 18-Mar-03

[6] CVE Editorial Board Members
http://www.cve.mitre.org/board/boardmembers.html
Last modified: 14-Feb-2003

[7] SecurityFocus Bugtraq Mailing list
http://www.securityfocus.com/archive/1
Accessed on: 22-Apr-2003

[8] SecurityFocus Bugtraq Searchable Database by Keyword
http://www.securityfocus.com/bid/keyword/
Accessed on: 15-Apr-2003

[9] Bugtraq ID Description and Classification
http://www.securityfocus.com/bid/6070/help/
Accessed on: 22-Apr-2003

[10] Bugtraq ID 6080 Microsoft WebDAV Denial of Service
http://www.securityfocus.com/bid/6070/info/
Accessed on 22-Apr-2003

[11] VulnXML Homepage
http://www.owasp.org/vulnxml/
Accessed on: 22-Apr-2003

---

[12] Open Web Application Security Project Homepage
http://www.owasp.org/
Accessed on: 22-Apr-2003

[13] VulnXML example: IIS Chunked Encoded Buffer Overflow
http://www.owasp.org/vulnxml/IISChunkedBO.xml
Accessed on: 22-Apr-2003

[14] ScanDo from KAVADO
http://www.kavado.com/ProductsScando.htm
Accessed on: 22-APR-2003

[15] WebScarab Homepage
http://www.owasp.org/webscarab/
Accessed on: 22-Apr-2003

[16] WebScarab Vision Document 1.0 7-Jul-2002
http://www.owasp.org/webscarab/VisionDocumentWebScarab.pdf
Accessed on: 22-Apr-2003

[17] WebSPHINX Homepage
http://www-2.cs.cmu.edu/~rcm/websphinx/
Accessed on: 22-Apr-2003

[18] Muffin Homepage
http://muffin.doit.org/
Accessed on: 22-Apr-2003

[19] OASIS homepage
http://www.oasis-open.org/home/index.php
Accessed on: 22-Apr-2003

[20] Symantec Enterprise Firewall HTTP URL Pattern Evasion Issue
Published 26-Mar-2003
http://www.securityfocus.com/archive/1/316309
Accessed on: 22-Apr-2003

[21] SNORT Open Source Network Intrusion Detection System Homepage
http://www.snort.org/
Page accessed an last updated: 22-Apr-2003

---

[22] Intrusion Detection Exchange Format (IDXF)
Internet Engineering Task Force Draft
http://www.ietf.org/internet-drafts/draft-ietf-idwg-beep-idxp-07.txt
Published October 22, 2002, expires April 22, 2003
Accessed on: 22-Apr-2003

[23] Intrusion Detection Message Exchange Requirements
Internet Engineering Task Force Draft
http://www.ietf.org/internet-drafts/draft-ietf-idwg-requirements-10.txt
Published October 22, 2002, expires April 22, 2003
Accessed on: 22-Apr-2003