# 从 OVAL 漏洞信息转换到 Nessus 扫描插件的研究与实现①

## Research and Implementation of Transforming from the Vulnerability Information of OVAL to Scanning Plugins of Nessus

王 猛 (燕山大学信息科学与工程学院 河北秦皇岛 066004)
(中科院高能物理研究所网络安全实验室 北京 100049)
刘必雄 (福建农林大学计算机与信息学院 350002)
许榕生 (中科院高能物理研究所网络安全实验室 北京 100049)
丁天昌 (燕山大学信息科学与工程学院 河北秦皇岛 066004)

**摘 要**:网络安全漏洞扫描应充分考虑执行的效率和信息的共享。本文通过对 OVAL 和 Nessus 的介绍,分析 OVAL Definition 与 Nessus plugin 之间的异同,提出并实现了一种将 OVAL 公布的漏洞信息转换成 Nessus 扫描插件的方法,共享了漏洞信息,并提升了漏洞检测的品质。最后经过实验说明该方法的可行性。

**关键词**:漏洞扫描 OVAL Definition Nessus plugin

## 1 引言

随着网络技术的发展,网络安全问题越来越受到人们的关注,全面检测一个网络的安全漏洞已经成为网络安全重要的参考依据。漏洞扫描就是用来检测目标系统是否存在着任何可能的漏洞,Nessus 是一款非常优秀的漏洞扫描工具,其强大的功能是依赖于其丰富的插件来实现的。[1] 虽然官方网站提供的插件很全面,但是安全漏洞层出不穷,为了保证 Nessus 不会遗失任何一个漏洞,必须共享其它组织提供的漏洞信息。而开放式漏洞评估标准语言 OVAL 提供了各种平台之间漏洞信息的交换。[2] 但跟 Nessus 不同的是,OVAL 本身没有适当的扫描程序,因为 OVAL 的主要目的是提升漏洞检测的质量。本文提出和实现了将 OVAL 公布的漏洞信息转换到 Nessus 的漏洞库中进行扫描的方案,从不同角度检测漏洞,以期提高漏洞发现的全面性和准确性。

## 2 OVAL 与 Nessus 介绍

### 2.1 OVAL 简介

CVE(Common Vulnerabilities and Exposures)是一个收集并命名各种操作系统上所有漏洞的计划。原来只是单纯字面描述,后来为了提高精确度以便让各种安全工具共享,便以制式规格加以描述、编号[3]。

OVAL(Open Vulnerability and Assessment Language)就是以 CVE 为基础,更详细的描述 CVE 的细节,以建立一系列漏洞检测的标准。CVE 及 OVAL 内容涵盖大多数信息安全相关软件工具、操作系统及专门的信息安全测试系统等。OVAL 提供了一个描述各平台安全漏洞的架构,可以透过指定条件搜寻,来判断目标系统有哪些安全漏洞。OVAL 最终目标在于强化漏洞评估测试的质量,让各种漏洞扫描工具共享其分析结果,增加测试的准确度。[4] OVAL 采用 XML 作为其内

① 基金项目:北京市教委科技发展计划项目(KM200610772006)

容编写规范，有利于 OVAL 整合进各种信息安全工具中。

### 2.2 OVAL definition 结构

OVAL 根据 MITRE[3] 规定的框架，以特定的 XML schema 标签编写各种 definition。在 Core Definition Schema 中定义一个 definition 最基本的轮廓。另外以三个 schema 文件编写 oval definition 必须了解的操作系统相关的信息：

表 1　Nessus plugin 的基本结构

| | |
|---|---|
| Script_id | 设置脚本的唯一标识 |
| Script_version | 设置脚本的版本号 |
| Script_name | 设置脚本的名称 |
| Script_description | 设置漏洞的详细描述信息 |
| Script_summary | 设置总结信息 |
| Script_category | 设置脚本的类别 |
| Script_copyright | 设置脚本的版权信息 |
| Script_family | 设置脚本所属的族 |
| Script_cve_id | 漏洞的CVE编号 |
| Script_bugtraq_id | 漏洞的bugtraq编号 |
| Script_require_ports | 设置使用该脚本所依赖的服务和端口 |
| Script_dependencie | 设置使用该脚本前要执行的其它脚本 |
| Script_timeout | 设置脚本的超时时间 |
| Attack code here | 攻击程序代码 |

（1）OVAL Definition Schema，编写 OVAL 漏洞定义、补丁的定义。

（2）OVAL Characteristics Schema，搜集有关于系统的资料，包含有系统设置的设定。Characteristics Schema 也定义了资料的交换格式，可以提供给漏洞扫描工具来应用。

（3）OVAL Results Schema，是 OVAL 评估系统的资料，它提供 OVAL 测试后以 XML 格式存储的结果，这个结果是目前评估系统的设定与 OVAL Definitions 的比较值[2]。

上述的 schema 还会根据不同的操作系统而产生独立的 schema，但内容都是继承上述三项基本的 schema。Definition 在编写过程中参考上面各项信息，来保证所有格式的固定。

### 2.3 Nsessus 简介

Nessus 是一款免费的漏洞扫描工具，其主要特点有：(1)Server/Client 的架构：Nessus 由服务器端和客户端组成，服务器负责执行扫描，客户端主要是一个 GUI 接口，用户通过客户端登入 Nessus 服务器，设置扫描策略并管理服务器端。(2)Plugin：Nessus 的各种检测程序都是以插件（Plugin）形式存在的，允许用户编写执行特定功能的插件，以进行更快速和更复杂的安全检查。(3)NASL：编写插件的主要语言，是针对 Nessus 特别开发的。对于安全性、速度及内存的用量都有相当好的表现，且易于移植到各平台。

虽然 Nessus 有提供更新 plugins 的功能，但有部分 plugins 一开始公布时是需要收费的，在公布后一周后才会提供给免费使用者[1][5]。

### 2.4 Nessus plugin 结构

一个完整的 Nessus plugin 由注册部分和攻击部分组成。注册部分包含加载该插件时所需要的描述信息；攻击部分包含用于攻击测试的代码。攻击完成后，可使用 security－warning（）和 security－hole（）函数报告是否存在相应安全漏洞。[6] 其基本结构如表 1 所示。

## 3　OVAL Definition 与 Nessus plugin 的差异

OVAL 用来描述一个系统的漏洞规范。OVAL 针对每一个检测过程都会对系统进行一连串的程序、档案版本的比对，是否存在特定名称的档案。根据每个漏洞预先定义好的条件一一分解直到变成一系列基本的动作。

Nessus Plugin 同样也分成注册和攻击两部分，注册部分也是在说明关于这份 plugin 所要检测的漏洞以及相关的 CVE 信息等。攻击则是实际的程序执行部分。

OVAL 的一个编写原则是尽量避免使用到真实利用漏洞的程序代码或只是判断漏洞补丁是否安装，而 Nessus 却是刚好相反的。对 OVAL 而言，这样检测过程不至于造成系统的负担，但检测结果可能不够精确。而对于 Nessus 来说，结果可以非常接近实际的漏洞核心，但越强烈的检测方式越有可能造成系统的负担。

所以 Nessus Plugin 与 OVAL 设计理念的最大差别是 Nessus Plugin 为了节省执行扫描时各方面的资源而采取共享函数库以及目标系统的各项信息。而 OVAL 却在每一个 definition 中详细描述各项信息，如果

单纯依照 definition 进行转换，可能会破坏 Nessus Plugin 的设计理念。因此在转换过程中首先要剔除 OVAL Definition 中的重复信息。

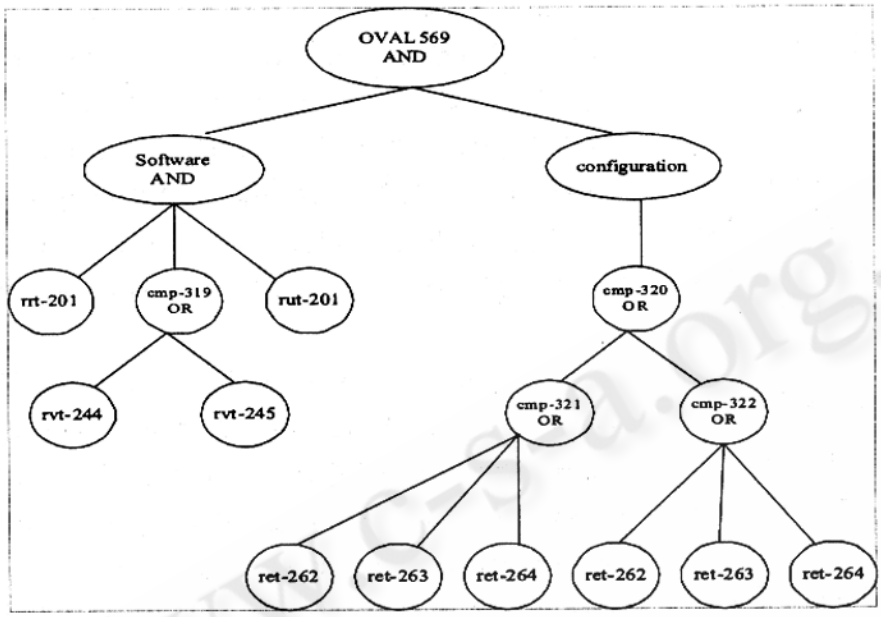


图 1　OVAL569 文件中特征条件所构成的多元树

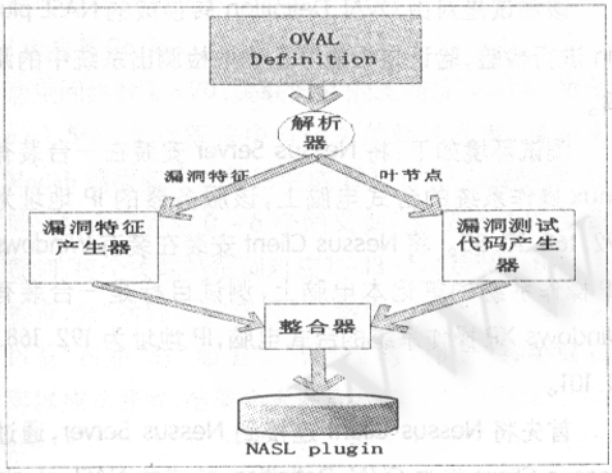


图 2　OVAL Definition 转换成 NASL plugin 的流程图

## 4　转换过程

OVAL 采用 XML 格式描述漏洞信息，而 XML 是用纯文本格式来表现结构化资料的一种方法。XML 中仅有一个元素称为根元素（Root element）或文件元素（document element），由该元素开始往下可以有更多的标签存在，因此，XML 文件可以用树状结构的方式来表示文件的内容，所以我们先对 XML 进行剖析，以便建立起 XML 文件所标示的树状结构。

在 OVAL 资料库中，对于一个漏洞的测试，初步分为软件和配置值两个方面，然后再分别针对软件和配置值，定义满足漏洞存在的条件，这样的定义方式，可以以树状结构建立它们彼此之间的关系。首先，由根（root）开始，下面有两个节点，分别是软件（software）和配置值（configuration），而这两个节点底下仍然有其他条件存在，这些条件还会由其他更多的条件组合而成，如此重复下去，会形成一棵多元树并且只有叶节点才是单一的测试条件。这样一棵多元树可以表示一个漏洞的关系式，本文将其称为逻辑树。

举 OVAL569[4] 为例子，图 1 表示由特征条件构成的 OVAL569 的多元树，Root 表示 OVAL569 文件，内容值为 AND，底下由 software 和 configuration 两大条件构成，成立与否由底下子节点决定，子节点间的关系存放在父节点上。每一个叶节点均对应着一个测试条件，如果成立，表示此节点的内容为 True，反之为 False. 只要能决定每一个叶节点的内容值，父节点的内容值也可以决定。如此反复进行，最后结果则存放在 root 里面。但是，在 OVAL 中这些测试条件都是分散在 XML 所表示的树状结构里，所以必须想办法追踪（trace）XML 所表示的树里面的各个节点，找出它们之间的关系，进而产生代表漏洞的逻辑树。最后，再针对树中每一个叶节点进行测试条件的操作，便可完成一个漏洞的判定。

首先从 OVAL 中过滤出必需的信息，因此先设计一

个解析器专门进行 XML 标签、属性和值等相关信息的抓取，由于 XML 使用树状结构的概念来安排标签，因此可以通过 Perl 现有的模块 DOM 来处理 XML 格式的 OVAL 文件，然后分成两大部分，一部分处理特征条件的逻辑关系的建立，称为漏洞特征产生器，另一部分负责处理逻辑树底下叶节点的，称为漏洞测试代码产生器。整个流程如图 2 所示。

### 4.1 漏洞特征产生器

漏洞特征产生器负责将每一个 OVAL Definition 文件测试条件之间的逻辑运算表达式找出来，其方法就是先建立逻辑树，再将它转换成逻辑运算表达式，然后只要能确定各个节点得值，就能判定是否为漏洞。其具体过程如下：

首先通过 Perl 的 DOM 模块建立一棵 XML 所标示的树，然后利用 trace 的方法找到需要的节点，由 root 开始从节点的第一个子节点开始往下，直到叶节点后再回到上一个节点选其他的子节点继续往下找，一直重复着这样的动作。先找到代表漏洞的两大条件 software 和 configuration，分别建立在逻辑树 root 下的左节点和右节点，两者的关系恒为 AND，存放在 root 节点内。然后先从 software 开始，往下找出底下子节点，同样子节点关系运算符存放在 software 底下，然后利用同样? 的方法，将这棵树由上而下，由左及右的生长直到全部的叶节点生长完毕，成为一棵具有漏洞特征的逻辑树。

但是树状结构并不能存放在文字形式的程序代码中，因此必须将建立起来的逻辑树转换成一种文字形式的逻辑运算表达式。本文采用的是类似二元树转成前序表达式的方法，从 root 开始，每一次都选择第一个子节点往下走，以此类推，直到叶节点后，再往上回到父节点并选择第二个子节点，然后再往下走到叶节点，重复上述步骤直到全部节点走完，即可产生由 AND、OR、NOT 运算符组成的逻辑运算表达式。例如图 1 里 OVAL569 漏洞特征的逻辑树经过上述方法转换后可以得到以下表达式：

AND(AND(rrt-201rut-201OR(rvt-244rvt-245))OR(OR(ret-262ret-263ret-264)OR(ret-265ret-266ret-267)))。

### 4.2 漏洞测试代码产生器

逻辑树的叶节点代表着单一的特征条件，每个条件都对应着一个测试，漏洞测试代码产生器的任务就是产生实际测试的程序代码，以决定每一个叶节点的值是 True 或 False，最后代入逻辑关系式来决定是否为漏洞。由于测试条件很多，因此将测试过程分成数个步骤，每一个步骤用一个函数表示，只要能确保函数的执行结果是正确的，也就能确定产生出来的程序代码所执行的结果是正确的。每一个测试类型所执行的步骤是相同的，只是检查的内容不同，在 Nessus 扫描工具中，它提供 NASL 语言让用户可以自行开发符合自己需求的扫描程序代码，其中有很多现成的函数可以使用。

由于 OVAL 系统地将每一种平台的漏洞分类，利用 NASL 现有的功能，就可以有效地完成 OVAL 的 XML 格式文件转换成程序代码。最后只要加上定期更新模块，负责固定时间下载新的 OVAL 漏洞信息，然后交给整合器产生 Nessus 扫描漏洞的程序代码，就可以完成从 OVAL Definition 到 NASL plugin 的自动转换。

## 5 实验结果

该测试是对由 OVAL Definition 转换成的 NASL plugin 进行检验，验证是否能够正确地检测出系统中的漏洞。

测试环境如下：将 Nessus Server 安装在一台装有 Linux 操作系统的台式电脑上，该服务器的 IP 地址为 192.168.10.80。将 Nessus Client 安装在装有 Windows XP 操作系统的笔记本电脑上，测试目标是一台装有 Windows XP 操作系统的台式电脑，IP 地址为 192.168.10.101。

首先将 Nessus Client 连接到 Nessus Server，通过 Nessus Client 将由 OVAL Definition 转成的 NASL plugin 脚本程序上传到 Nessus Server。然后勾选要检测的项目，输入被测对象的 IP 地址。最后进行实际地测试。得到如图 3 所示的测试结果。

该结果显示被测对象的系统出现两个严重的漏洞和两个警告，需要下载漏洞补丁，防止黑客的入侵。

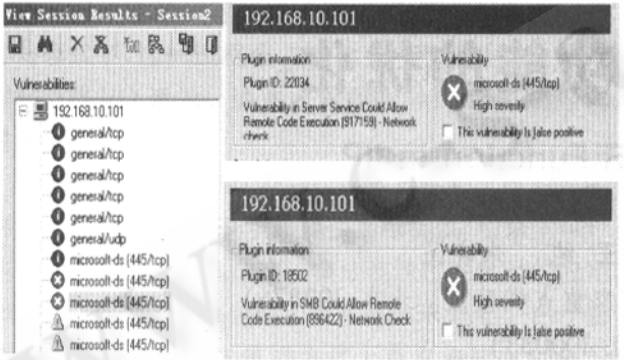


图 3 测试结果

## 6 结束语

全面检测网络安全漏洞不仅需要强大的扫描工具,而且还要有完善的漏洞库的支持。然而,Nessus 的漏洞插件库很难及时更新其它安全工具发现的漏洞信息,本文提出和实现的 Nessus 分享 OVAL 漏洞信息的方案,解决了 Nessus 的不足之处,并且能够提升漏洞检测的质量,提高发现漏洞的精确度。

## 参考文献

1 RENAUD DERAISON. Nessus network auditing[M]. Syngress Publishing ? 2004 (508 pages).

2 M. WOJCIK, D. PROULX, J. BAKER, R. ROBERGE. Introduction to OVAL: A language to determine the presence of computer vulnerabilities and configuration issues. The MITRE Corporation, 2005.

3 Common Vulnerabilities and Exposures URL: http://cve. mitre. org/.

4 OVAL URL: http://oval. mitre. org/

5 Nessus Vulnerability Scanner URL: http://www. nessus. org/index. php.

6 肖晖、张玉清,Nessus 插件开发及实例[J],计算机工程,2007 年 1 月第 33 卷第 2 期,241 -243.

7 SCAMBRAY J, MCCLURE S, KURTZ G, 网络安全机密与解决方案[M],杨继张译,北京:清华大学出版社,2000.