УДК 004.02

Н. И. МАЗУРЕНКО, В. С. ХАРЧЕНКО, А. В. ГОРБЕНКО

Национальный аэрокосмический университет им. Н. Е. Жуковского «ХАИ», Украина

ДИНАМИЧЕСКАЯ РЕКОНФИГУРАЦИЯ ВЕБ-СИСТЕМЫ НА ОСНОВЕ МЕТРИЧЕСКОГО АНАЛИЗА БАЗ ДАННЫХ УЯЗВИМОСТЕЙ OTS-КОМПОНЕНТОВ

В статье проанализированы существующие методы анализа уязвимостей компонентов. Предложена методика сравнительной оценки конфигураций программных систем на основании данных об уязвимостях компонентов и связанных уязвимостей. Разработана архитектура веб-сервиса для динамического конфигурирования программной системы. Предложена методика оценки выигрыша в безопасности при динамической реконфигурации. Дан реальный пример получаемого эффекта при реконфигурации операционных систем Ubuntu linux 12.04, Microsoft Windows Server 2008 SP2 x64 и Mac OS X Server 10.7.4 с учетом их уязвимостей.

Ключевые слова: *уязвимость, безопасность, метрика, реконфигурация, CVSS*

Введение

Разнообразие (диверсность) технологий является одним из наиболее эффективных способов обеспечения отказоустойчивости программного обеспечения [1]. Имея многоверсионную архитектуру, устойчивую к уязвимостям, можно значительно повысить уровень безопасности (как функциональной, так и информационной). Для обеспечения информационной безопасности необходимы процедуры достоверного оценивания устойчивости конфигурации к вторжениям и динамическое развертывание наиболее устойчивой актуальной конфигурации.

Существует множество публичных баз данных уязвимостей, которые предоставляют большой объем данных для анализа безопасности компонентов. Из них можно выделить 3 основных: National Vulnerability Database (NVD), Common Vulnerabilities and Exposures (CVE) и Open Sourced Vulnerability Database (OSVDB) [2], предоставляющие информацию в виде XML файлов, которые можно проанализировать без предварительного разбора входных данных.

Целями такого анализа являются оценивание (фиксация) различных параметров уязвимости для получения информации, позволяющей судить о надежности (безопасности) компонента.

Самой распространенной методикой оценивания тяжести уязвимости является Common Vulnerability Scoring System (CVSS) v2, которая предоставляет теоретическую и весьма приближённую оценку уязвимости [4]. На основе данной оценки и информации о так называемых связанных уязвимостях (проявляющихся только при наличии двух или более фиксированных компонентов) следует усовершенствовать методику определения параметров уязвимостей, в части учета конфигураций компонентов для динамической реконфигурации архитектуры.

В известных работах, в частности [1], такая реконфигурация предусматривается, однако поддерживающие механизмы детально не проработаны.

Цель статьи – разработать архитектуру сервиса, позволяющего производить динамическую реконфигурацию на основе метрических оценок уязвимостей, включая связанные уязвимости компонент.

1. Анализ методики оценивания CVSS

Данная методика включает 3 основных группы метрик [3] (Рис.1).

Базовые метрики (Base): используются для описания основополагающих сведений об уязвимости (возможности эксплуатации уязвимости и влиянии уязвимости на систему).

Временные метрики (Temporal): при вычислении метрики учитывается время, например, опасность уязвимостей снижается с выходом официального обновления безопасности.

Контекстные метрики (Environmental): вопросы контекста принимаются во внимание при оценке опасности уязвимости. Например, чем больше компонент (и систем) подвержены уязвимости, тем выше ее опасность.

На первом этапе проводится расчет оценки базовых метрик, их объединение в единую оценку, на которую уже затем накладываются временная и контекстная оценки. Учитывая то, что данные фор-

мируются аналитиками бюллетеней уязвимостей, производителями продуктов в области информационной безопасности или производителями приложений метрики могут заполняться неточно.

Рис. 1. Схема групп метрик CVSS

Так же градации метрик может быть недостаточно, что приведет к неправильной конечной оценке. К примеру, уязвимость в пути веб-приложения будет оценена как (AV:N/Au:N/AC:L/C:P/I:N/A:N) с общей оценкой 5,0. Уязвимость, которая позволяет атакующему проникнуть в файловую систему и прочитать любой файл, доступный для веб-сервера, будет оценен как (AV:N/Au:N/AC:L/ C:P/I:N/A:N) с такой же общей оценкой 5,0 [4]. Эти две уязвимости представляют собой абсолютно разные угрозы, но по стандарту CVSS v2 они будут иметь одинаковые вес, что никак не соответствуют действительности.

2. Использование статистики существующих «эксплоитов»

Чтобы знать в действительности, насколько часто атакуется та и или иная уязвимость в программном продукте, можно воспользоваться статистикой количества «эксплоитов» определенной уязвимости. Одним из самых известных и публично доступных архивов, который хранит такую информацию, является ExploitDB. Он хранит сейчас данных более, чем по 27500 «эксполоитам» [5]. К примеру, возьмём уязвимость CVE-2009-2698, которая имеет 3 зарегистрированных «эксплоита», основанных на ошибке в ядре операционной системы Linux. Общая оценка данной уязвимости в соответствии с методикой CVSS равна 7, однако, если учесть наличие «эксплоита», уровень ее опасности следует увеличить. В зависимости от количества найденных «эксплоитов» и среднего количества S в базе данных (максимально 8), вычисленного с использованием методики CVSS, можем записать:

$$PS = S + 0,2 \cdot E, \qquad (1)$$

где PS *(product severity)* – уровень опасности уязвимости, E – количество подтвержденных «эксплоитов».

3. Оценка конфигурации компонентов программной системы

В качестве программной системы возьмем сервис-ориентированную архитектуру (СОА), которая является типичным представителем многоуровневых архитектур. СОА является программной системой, предназначенной для обеспечения сетевого взаимодействия между компьютерами. СОА предполагает наличие трех участников: поставщика сервиса, потребителя сервиса и реестра сервисов (рис. 2) [6].

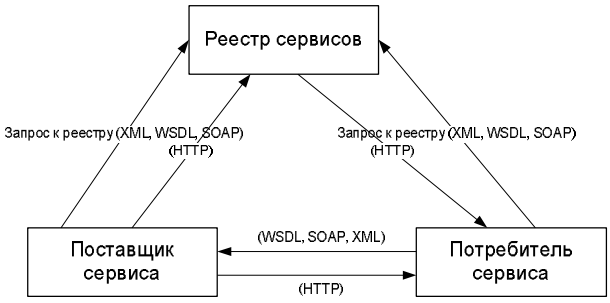

Рис. 2. Общая схема СОА

В архитектуре провайдера можно выделить различные уровни. На примере веб-сервиса такими уровнями будут: аппаратный уровень, уровень операционной системы (ОС), уровень веб-сервера, уровень приложений и уровень баз данных (БД).

Программный уровень представляет собой цепочку компонентов. Оценка безопасности конкретного компонента программной системы, с учетом статистики «эксплоитов», может быть записана как:

$$VLC_j = \sum_{i=1}^{n} PS_i, \qquad (2)$$

где VLC – уровень опасности компонента системы j-го компонента, n – количество найденных уязвимостей для текущего компонента, PS_i – уровень опасности i-той уязвимости. Следовательно, формулу для оценки программной системы можно записать в виде:

$$VLS = \sum_{i=1}^{k} VLC_k. \qquad (3)$$

Теперь, учтем, что система может иметь связанные уязвимости, т.е. такие уязвимости, которые появляются только в случае использование двух определенных компонентов (4).

$$VLS = \sum_{i=1}^{k} VLC_k + \sum_{m=1}^{p} VLAC_m, \qquad (4)$$

где $VLAC_m$ – оценка уровня опасности m-той связанной уязвимости, p – количество связанных уязвимостей системы.

Проведем расчет. За основу возьмем конфигурацию, описываемую таблицей 1

Таблица 1

Конфигурация ПС

Операционная система	RedHat Linux Enterprise ver 5.0 (desktop workstation)
Веб-сервер	Apache HTTP Server 2.2.3
Приложение	PHP 5.3.0
База Данных	MySQL 5.1.23

Исходя из конфигурации систем, найдем все уязвимости каждого компонента (табл. 2), а так же проверим наличие связанных уязвимостей. В данной конфигурации нашлась только одна уязвимость CVE-2006-5752 для связки ОС и Веб-сервера с оценкой CVSS 4; «эксплоитов» не обнаружено.

Таблица 2

Уязвимости и «эксплоиты» выбранной программной системы

Компонент	Уязвимость	Оценка CVSS	Количество «эксплоитов»
Операционная система	CVE-2007-0771	5	0
	CVE-2007-1351	9	0
Веб-сервер	CVE-2010-0425	10	1
	CVE-2013-1862	5	0
	CVE-2009-3555	6	3
	CVE-2013-2249	8	0
Приложение	CVE-2013-4248	4	0
	CVE-2013-1643	5	0
	CVE-2009-4018	8	1
	CVE-2013-1824	4	0
	CVE-2012-2376	10	0
	CVE-2013-4113	7	0
База данных	CVE-2010-2008	4	1
	CVE-2013-0389	7	0
	CVE-2014-0393	3	0

С учетом формул (1-4) имеем:

$$VLS = 14 + 29,8 + 38,2 + 14,2 + 4 = 100,2.$$

Получив конечную оценку, можно сравнивать конфигурации, и делать вывод о степени безопасности использования программной системы в реальных условиях.

3. Динамический пересмотр информационной безопасности конфигурации

Для того, чтобы разработать систему, устойчивую к вторжениям, необходим постоянный обзор версий программных продуктов по мере их появления, анализ новых уязвимостей и пересмотр возможных конфигураций для программной системы.

В первую очередь необходимо ежедневное обновление баз данных уязвимостей от всех внешних источников, а так же пересмотр наличия «эксплойтов» в обновленной базе данных. Как только актуальная информация получена, следует проверить все сохраненные конфигурации, которые сравнивались пользователями. Пересчет значений оценки информационной безопасности конфигурации должен проходить с учетом возможной альтернативы конфигурации, учитывая новые версии продуктов, использовавшихся при предыдущем сравнении.

Если финальный результат сравнения оценок информационной безопасности будет значительно отличаться от предыдущего сравнения, то необходимо провести реконфигурацию программной системы. Для активации конфигурации нужно подготовить сценарий для развертывания и запустить его на экземпляре выбранной ОС. Схема архитектуры динамического реконфигурирования представлена на Рисунке 3. В качестве примера такой реконфигурации можем рассмотреть 3 конфигурации для конкретной архитектуры - веб-сервера (таблица 3).

Таблица 3

Список конфигураций

Номер конфигурации	ОС	Веб-сервер	Приложение	БД
1	Ubuntu linux 12.04	Apache HTTP Server 2.2.24	PHP 5.3.0	MySQL 5.5.27
2	Microsoft Windows Server 2008 SP2 x64	Microsoft IIS 7.0	Microsoft ASP.NET 3.5	Microsoft SQL Server 2008 x64
3	Mac OS X Server 10.7.4	Nginx 0.7	Oracle Weblogic 10.3	Oracle Database Server 11

На рисунке 4 представлен график изменения метрики опасности данных конфигураций в течение определенного периода времени. При выборе в качестве исходной конфигурации $M_{конф\ j}$ рассмотрим все конфигурации на момент t_1. Показатель системы на момент t_1 можно представить как:

$$M_{сист}(t, t_1) = \min_i \{M_{ki}(t, t_1)\}, \qquad (5)$$

где i – номер конфигурации.

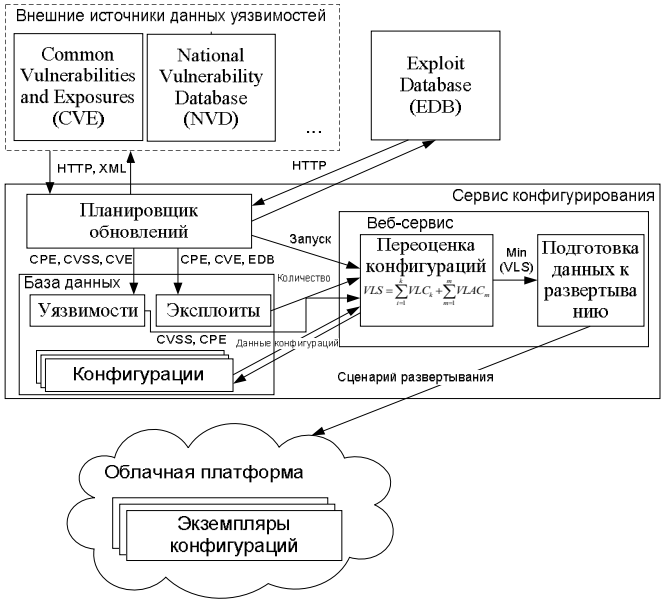

Рис. 3. Общая схема динамического обновления сравнений конфигураций

Рис. 4. График изменения метрики опасности, рассчитанной по формуле (4)

Тогда выигрыш показателя на момент t_1 определим как

$$\Delta M(t, t_1) = M_{kj}(t, t_1) - \min_i \{M_{ki}(t, t_1)\}. \quad (6)$$

Наиболее безопасной является конфигурация 3, т.к. имеет минимальный показатель M_k. В моменты времени t_2 и t_3 можем определить наилучшее значение параметров $M_{kj}(t,t_2)$ и $M_{kj}(t,t_3)$ соответственно. В момент t_2 наилучшей будет конфигурацией 2, а t_3 – 3.

Приращение в безопасности системы в целом можно рассчитать как:

$$\Delta M(t) = \Delta M(t, t_1, t_2) + \Delta M(t, t_2, t_3) + \\ + \Delta M(t, t_2, > t_3). \quad (7)$$

Так как $\Delta M(t, t_1, t_2)$ и $\Delta M(t, > t_3)$ равны нулю, при выборе исходной конфигурации 3, то

$$\Delta M(t) = \Delta M(t_2, t_3) = \\ = M_{k3}(t, t_2, t_3) - M_{k2}(t, t_2, t_3). \quad (8)$$

Данная область заштрихована на рисунке 4 и представляет собой выигрыш информационной безопасности в случае выбора данных по трем конфигурациям в период с 01.01.2013 по 31.12.2013.

Заключение

Динамический пересмотр конфигураций дает возможность получать актуальную оценку информационной безопасности как уже ранее оцененных и использованных конфигураций, так и альтернатив-

ных конфигураций с новыми версиями продуктов, полученных при обновлении БД.

Динамическое реконфигурирование повышает устойчивость к вторжениям, учитывая уязвимости каждого из компонентов. Уточнение оценки за счет существующих «эксплоитов» в уязвимостях поддерживает корректировку популярности использования уязвимости. Это делает оценку более практичной для использования в реальных динамично изменяющихся условиях. Выигрыш информационной безопасности может быть значительным и уменьшить вероятность вторжений в несколько раз.

Дальнейшие исследования целесообразно направить на анализ различных процедур реконфигурации веб-систем при использовании принципа диверсности.

Литература

1. Intrusion-avoiding architecture making use of diversity in the cloud-based deployment environment [Text]/ A. Gorbenko, V. Kharchenko, O. Tarasyuk, A. Romanovsky // Newcastle University, Technical Report Series, No. CS-TR-1262. – July, 2011. – 18 p.

2. Моделирование гарантоспособных систем и сетей. Лекционный материал [Текст]/ под ред. В. С. Харченко. – Х. : Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», 2008. – С. 137-138.

3. A complete guide to the common vulnerability scoring system Version 2.0 [Electronic resource]. – Available to: http://www.first.org/cvss/cvss-guide.html

4. The CVSSv2 Shortcomings, faults, and failures formulation [Electronic resource]. – Available to: http://www.riskbasedsecurity.com /reports/CVSS-ShortcomingsFaultsandFailures.pdf.

5. CVE Reference map for source EXPLOIT-DB [Electronic resource]. – Available to: http://cve.mitre.org/data/refs/refmap/source-EXPLOIT-DB.html

6. Web services architecture [Electronic resource]. – Available to: http://www.w3.org/TR/ws-arch/

Поступила в редакцию 15.02.2014, рассмотрена на редколлегии 24.03.2014

Рецензент: д-р техн. наук, проф. В. А. Заславский, Киевский национальный университет им. Тараса Шевченко, Киев, Украина.

ДИНАМІЧНЕ РЕКОНФІГУРУВАННЯ ВЕБ-СИСТЕМИ ШЛЯХОМ МЕТРИЧНОГО АНАЛІЗУ БАЗ ДАНИХ ВРАЗЛИВОСТЕЙ OTS-КОМПОНЕНТІВ

М. І. Мазуренко, В. С. Харченко, А. В. Горбенко

У статті проаналізовано існуючі методи аналізу уразливостей компонентів. Запропоновано методику оцінки програмних систем на підставі даних про уразливості компонентів та зв'язані уразливості конфігурацій. Розроблено архітектуру веб-сервісу для динамічного конфігурування програмної системи. Запропоновано методику оцінку виграшу в безпеці при динамічній реконфігурації. Надано приклад одержуваного ефекту при реконфігурації операційних систем Ubuntu linux 12.04, Microsoft Windows Server 2008 SP2 x64 і Mac OS X Server 10.7.4 з урахуванням їх вразливостей.

Ключові слова: вразливість, безпека, метрика, реконфігурація, CVSS.

WEB- SYSTEM DYNAMICAL RECONFIGURATION BASED ON METRIC ANALYSIS OF VULNERABILITY DATABASES OTS-COMPONENTS

M. I. Mazurenko, V. S. Kharchenko, A. V. Gorbenko

Existing methods for determining the vulnerability components have been analyzed. A method of the comparative evaluating of software system configurations based on vulnerability components and interconnection vulnerabilities is suggested. The architecture of a web service to dynamically configure the software system is developed. The techniques of security assessment for dynamic reconfiguration are proposed. A real example of the resulting effect in the reconfiguration of operating systems Ubuntu linux 12.04, Microsoft Windows Server 2008 SP2 x64 and Mac OS X Server 10.7.4 is given with regard to their vulnerabilities.

Key words: vulnerability, security, metric, reconfiguration, CVSS.

Мазуренко Никита Игоревич – магистрант кафедры компьютерных систем и сетей Национального аэрокосмического университета им. Н. Е. Жуковского «ХАИ», Харьков, Украина, e-mail: mnk.ami@gmail.com.

Харченко Вячеслав Сергеевич – д-р техн. наук, профессор, зав. каф. компьютерных систем и сетей Национального аэрокосмического университета им. Н. Е. Жуковского «ХАИ», Харьков, Украина, e-mail: v_s_kharchenko@ukr.net.

Горбенко Анатолий Викторович – д-р техн. наук, профессор, профессор каф. компьютерных систем и сетей Национального аэрокосмического университета им. Н. Е. Жуковского «ХАИ», Харьков, Украина, e-mail: anatoliy.gorbenko@gmail.com.