ПРОБЛЕМЫ УЯЗВИМОСТИ WEB И СРЕДСТВА ДЛЯ АНАЛИЗА БЕЗОПАСНОСТИ WEB-ПРИЛОЖЕНИЙ

Автор: **Ильенко Ф.В., Приходько Т.А.**
Источник: Інформаційні управляючі системи та комп'ютерний моніторинг (ІУС КМ - 2013) - 2013 / Матеріали ІІІ міжнародної науково-технічної конференції студентів, аспірантів та молодих вчених. — Донецьк, ДонНТУ — 2013

Аннотация

Ильенко Ф.В., Приходько Т.А. «*Средства для анализа безопасности WEB-приложений*». *В статье выполнен анализ уязвимостей Интернет-приложений зарубежных и отечественных производителей, в том числе в зависимости от языка разработки. Проведена классификация самых распространенных уязвимостей. Приведен обзор средств для тестирования безопасности web-приложений.*

Ключевые слова: *безопасность, веб-приложения, типы атак, межсайтовый скриптинг, Injection.4ewe4ko@mail.ru*

Постановка проблемы

Web-приложения являются одними из наиболее небезопасных систем на сегодняшний день. Чем больше критически важных и конфиденциальных данных хранит программное обеспечение, тем важнее становится проведение аудита его безопасности. Уязвимости веб-приложений выливаются в кражи миллионов номеров кредитных карт, потери миллиардов долларов США ежегодно.

Тестирование безопасности - это тип тестирования, основное предназначение которого – убедится, что конфиденциальные данные останутся конфиденциальными, а пользователь системы сможет сделать только то, что ему положено по правам доступа. Оценка защищенности Web-приложений может выполняться как с использованием методики "черного ящика", так и путем анализа исходных кодов. Второй способ более эффективен, но и более трудоемок. Метод "черного ящика" заключается в проведении работ по оценке защищенности информационной системы без предварительного получения какой-либо информации о ней со стороны владельца. Метод "белого ящика" заключается в том, что для оценки защищенности информационной системы используются все необходимые данные о ней, включая исходный код приложений.

С развитием Интернета появляются как положительные, так и отрицательные моменты. Растет количество людей и способов, чьей целью является нарушение работы ресурса, кража информации и прочие подобные действия. В последнее время очень высокое внимание уделяется вопросу об устойчивости сайтов к различным рода атакам.

Актуальность этой проблемы порождает множество проектов, призванных помочь разработчикам веб-приложений повысить надёжность своих продуктов. Фактически, тестирование веб-приложения на наличие в нём уязвимостей является прекрасной превентивной мерой, поскольку позволяет определить недостатки приложения ещё на этапе разработки и закрытого тестирования, тем самым, выдав на выходе надёжный продукт.

Цели

Основной целью данной работы является изучение средств безопасности web-приложений, а так же определение наиболее распространенных уязвимостей, поиск методов защиты и средств анализа безопасности web-приложений.

ТОП-10 наиболее распространенных уязвимостей

В данной статье приведен анализ ТОП-10 уязвимостей по результатам исследований российских специалистов и международной организации по обеспечению безопасности WEB-приложений (OSWAP).

Перове исследование было проведено в рунете экспертами российской компании Positive Technologies в период с 2010 по 2011 год. Исследование проводилось в рамках оказания услуг по тестированию на проникновение и анализу защищенности информационных систем. Детальный анализ веб-приложений ключевых российских компаний и предприятий (объектами тестирования стали 123 сайта) дал следующие результаты [2]: было обнаружено всего 1817 уязвимостей различной степени риска.

Десять уязвимостей, выявленных на наибольшем количестве сайтов, представлены на рис. 1. Лидер в этом списке — *Cross-Site Request Forgery (CSRF)*, которой оказался подвержен 61% всех проверенных сайтов. CSRF (или Подделка HTTP-запросов) основана на том, что все браузеры, когда запрашивается какая-то страница сайта, посылают на сервер вместе с запросом куки-файлы, содержащие идентификаторы сессий и другую информацию, связанную с этим сайтом. Поэтому есть возможность подделать URL так, что он будет выполнять какое-либо действие на атакуемом сервере, и браузер не будет знать, что данный URL не затребован самим пользователем. Сервер же обработает подделанный URL так, как будто это есть корректный URL, который запросил сам пользователь.

Далее следуют *Information Leakage u Brute Force* — 54% и 52% сайтов, а также *SQL Injection* с критическим уровнем риска, обнаруженная на 47% ресурсов. На более низких позициях в список входят еще две уязвимости с высокой степенью риска — *OS Commanding* и *Path Traversal* (обе с долей

уязвимых сайтов в 28%). В список десяти самых распространенных уязвимостей вошли также недостатки среднего уровня риска: *Insufficient Anti-automation* (42% исследованных сайтов), *Cross- Site Scripting* (40%), *Predictable Resource Location* (36%) и *Insufficient Transport Layer Protection* (22%). В целом минувший 2011 год был отмечен ростом доли ресурсов, подверженных уязвимости Cross-Site Request Forgery, тогда как SQL Injection и OS Commanding стали встречаться реже, чем в 2010 г.

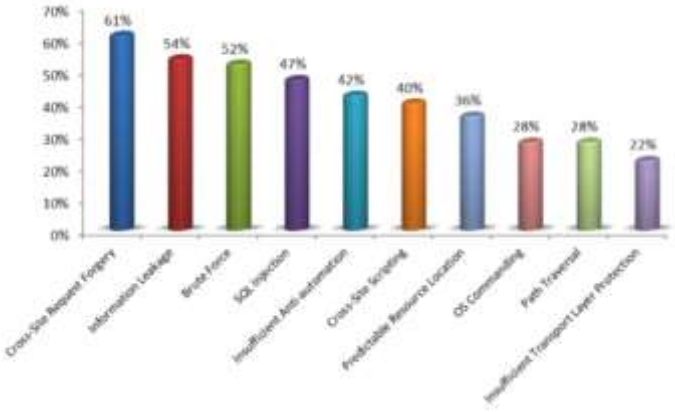

Рис.1 Наиболее распространенные уязвимости (доля сайтов, %) [1,2].

И в целом уязвимости с высокой степенью критичности чаще встречаются на PHP сайтах: 81% ресурсов подвержены подобным проблемам.

По результатам OSWAP (сравнение по результатам 2010 и 2013г.) можно привести следующие данные (рис.2).

OWASP Top 10 – 2010 (Previous)	OWASP Top 10 – 2013 (New)
A1 – Injection	A1 – Injection
A3 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A2 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References	A4 – Insecure Direct Object References
A6 – Security Misconfiguration	A5 – Security Misconfiguration
A7 – Insecure Cryptographic Storage – Merged with A9 →	A6 – Sensitive Data Exposure
A8 – Failure to Restrict URL Access – Broadened into →	A7 – Missing Function Level Access Control
A5 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
<buried in A6: Security Misconfiguration>	A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards	A10 – Unvalidated Redirects and Forwards
A9 – Insufficient Transport Layer Protection	Merged with 2010-A7 into new 2013-A6

Рис.2. ТОП-10 уязвимостей по данным OSWAP[3].

В этом исследовании на первом месте стоят Инъекции, Кроссайтовый скриптинг в 2013г. переместился со второго на третье место, а кроссайтовые запросы сместились с 5-го на восьмое место. Следует заметить, что отличия в российских и зарубежных рейтингах уязвимостей (на примере CSRF) связано с тем, что CSRF лидировал в OWASP Топ-10 в течение 6 лет, и организации и разработчики структуры сфокусировались на этом достаточно, чтобы значительно сократить число уязвимости CSRF в реальных мировых приложениях, тогда как отечественные сайтостроители только начали борьбу.

В целом, ключевые факторы эволюции зарубежного списка уязвимостей (рис.2) следует связывать в первую очередь с прогрессом в развитии атак, сделанным нападавшими, а так же с появлением новых технологий борьбы с новыми уязвимостями, также как и более строгих методов защиты наряду с развертыванием все более и более сложных систем.

Уязвимости, характерные для различных языков программирования

Среди тестируемых ресурсов встречаются веб-приложения, написанные на различных языках программирования; при этом для каждого языка характерен свой набор наиболее значимых уязвимостей (по данным исследования группы Positive Technologies [1]. Большинство разработчиков предпочитают PHP: на нем написаны 63% всех протестированных сайтов. Значительны доли ASP.NET (19%) и Java (14%). Остальные языки программирования встречаются гораздо реже. Распределение сайтов — участников тестирования по лежащему в их основе языку программирования визуально представлено на рис. 3.

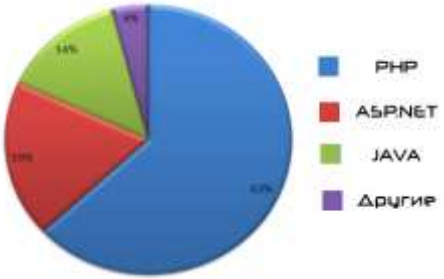

Рис. 3. Статистика использования языков программирования при создании динамических Web-страниц [1])

Безопасность веб-приложений действительно развивается с постоянными изменениями с методами и технологиями атак, как сказал Иеремия Гроссман, основатель и технический директор компании WhiteHat Security. «На этот раз мы хотели сосредоточиться на языках программирования, поскольку именно оттуда все и начинается. Если организации имеют лучшее представление о

том, как они используют язык тарифа в этой области, они могут быть более бдительными в течение всего жизненного цикла разработки, и мы надеемся избежать больших проблем в будущем».

Среди веб-приложений, разработанных на языке PHP, наиболее часто встречаются следующие уязвимости: Cross Site Scripting, Cross-Site Request Forgery, Insufficient Anti-automation, а также две критические — SQL Injection и Path Traversal. Для платформы ASP.NET характерны те же самые уязвимости (исключение составляет Application Misconfiguration, занявшая место Path Traversal). В рейтинг среды Java вошли уязвимости Insufficient Authorization, Cross-Site Request Forgery, Application Misconfiguration, Insufficient Authentication и лишь одна критическая уязвимость — OS Commanding (рис 3.). Можно заметить, что доли уязвимых сайтов на ASP.NET и Java ниже по сравнению с аналогичными показателями для языка PHP (см. рис 4).

PHP	Доля сайтов, %	ASP.NET	Доля сайтов, %	Java	Доля сайтов, %
Cross-Site Request Forgery	73	Cross-Site Scripting	39	Insufficient Authorization	41
SQL Injection	61	Cross-Site Request Forgery	35	Cross-Site Request Forgery	35
Cross-Site Scripting	43	Insufficient Anti-automation	35	Application Misconfiguration	29
Insufficient Anti-automation	42	SQL Injection	22	Insufficient Authenticatio	29
Path Traversal	42	Application Misconfiguration	17	OS Commanding	29

Рис.4. Наиболее распространенные уязвимости в зависимости от языка программирования [4]

Средства анализа безопасности Web-приложений

Защита уязвимых Web-приложений может осуществляться либо путем устранения уязвимых мест в Web-приложении, либо с помощью специализированных средств защиты Web-приложений. Слабые места Web-приложения можно устранить посредством тщательного анализа исходного кода приложения на предмет ошибок либо периодически выполняя тесты на проникновение.

В настоящее время известны следующие способы обнаружения уязвимостей в web-приложениях

1. Тестирование функций
    - Метод «черного ящика» (black-box)
    - Метод «серого ящика» (gray-box)

- Метод «белого ящика» (whitebox)

2. Анализ исходного кода
   - Статический анализ (без реального исполнения программы)
   - Динамический анализ (отладка в процессе исполнения)
   - Ручной анализ
3. Фаззинг (fuzzing) (на вход программы подаются невалидные, непредусмотренные или случайные данные.)
4. Бинарный анализ приложения (binary analysis)

Первые две группы можно охарактеризовать как [6]: 1) методы, анализирующие работу развернутого на стенде web-приложения без обращения к исходным кодам web-приложения; 2) методы, анализирующие исходные коды web-приложения и конфигурационные настройки.

Группа 1) рассматривает web-приложение с точки зрения внешнего пользователя, то есть потенциального злоумышленника. В эту группу входят следующие методы:

I.1. Метод получения идентифицирующей информации о web-приложении и выявления его уязвимостей с помощью бюллетеней безопасности (security advisory).

I.2. Метод тестирования на проникновение.

Метод получения идентифицирующей информации (I.1) получил широкое распространение в виду его простоты и доступности. Основан он на анализе HTTP ответов WEB-сервера проведении атак ввиду своей простоты и доступности. Метод не позволяет найти новые уязвимости web-приложения, а его полезность для обнаружения уязвимостей новых приложений состоит в том, что он дает возможность определить саму **возможность** утечки информации о web-приложении.

Метод тестирования на проникновение (penetration testing) предусматривает тестирование работающего на стенде web-приложения путем посылки запросов, которые эмулируют пользовательскую активность (в том числе, и некорректные запросы, соответствующие действиям злоумышленника). При этом считается, что злоумышленник обладает такими же возможностями, как и обычный пользователь, т.е. не имеет доступа к исходным кодам web-приложения, конфигурационным настройкам и т.п. Задача – получение и анализ структуры WEB-приложения. По структуре web-приложения проверяются типовые уязвимости, связанные с неправильным конфигурированием web-приложения и web-сервера.

Группа 2) состоит из следующих методов:

II. 1. Метод статического анализа исходных кодов web-приложения.

II. 2. Метод динамического анализа исходных кодов web-приложения.

II. 3. Анализ вручную.

Для обнаружения уязвимостей методом статического анализа используется два основных подхода: анализ типов безопасности [6] и анализ потоков данных.

Поиск методом статистического анализа исходного кода основан на использовании сигнатур, которые в свою очередь базируются на аппарате регулярных выражений. В силу своей простоты метод получил наибольшее распространение. При использовании статического анализа неизбежны ошибки первого и второго рода, когда анализирующее приложение не сможет выявить некоторое число уязвимостей в силу отсутствия соответствующей сигнатуры. Возможны также многочисленные ложные уведомления о наличии уязвимости.

Динамический анализ более эффективен по сравнению со статическим. Программное средство, позволяющее выполнить динамический анализ кода, досконально разбирает синтаксис языка программирования, на котором написано исследуемое приложение, и проводит ряд проверок, направленных на выявление необработанных данных со стороны пользователя, поступающих в потенциально опасные функции приложения в качестве аргументов. Такой анализ позволяет за короткое время выявить все грубые ошибки, допущенные программистами, и выдать отчет с минимальным количеством ошибок первого и второго рода. Однако из-за сложности реализации таких средств, а также в силу необходимости поддержки множества языков программирования (PHP, ASP, Perl, Python, Java и др.), на которых могут быть написаны Web-приложения, подобных средств не так уж и много.

Выводы.

Проведена классификация самых распространенных уязвимостей. Результаты показали, что среди выявленных уязвимостей на наибольшем количестве сайтов лидером является — Cross-Site Request Forgery (CSRF).

В работе был проведен анализ средств безопасности web-приложений. В качестве метода для дальнейшего развития исследований в магистерской работе решено работать с методами, анализирующими исходные коды web-приложения и конфигурационные настройки.

Список литературы

1. Positive Technologies – безопасность,консалтинг/ Интернет-ресурс. - Режим доступа: www/ URL: http://www.ptsecurity.ru/ - Загл. с экрана.
2. С.Гордейчик, Д.Евтеев и др. "Статистика веб-уязвимостей за 2010-2011 годы" Интернет-ресурс. - Режим доступа: http://ptsecurity.ru/lab/analytics/
3. "OSWAP Top10 -2013 rcl. The Ten Most Critical Web Application Security Risk" Интернет-ресурс. - Режим доступа: http://oswaptop10.googlecode.com
4. Издательство БХВ-Петербург, Тактика защиты и нападения на Web-приложения – 2005. – 432с.

5. Уязвимости | информационный портал о безопасности/Интернет-ресурс. - Режим доступа: www/ URL: http://www.securitylab.ru/vulnerability/ - Загл. с экрана.
6. Козлов Д. Д., Петухов А. А. "Методы обнаружения уязвимостей в web-приложениях" / Программные системы и инструменты: тематический сборник ф-та ВМиК МГУ им. Ломоносова N 7. П/р Л.Н. Королева. М: Издательский отдел ВМиК МГУ. Изд-во МАКС Пресс, 2006 г.