УДК 004.42

В. В. Степчик, Д. Д. Узун, Ю.А. Узун

Особенности автоматизации процесса тестирования уязвимостей web-ресурсов с использованием ПО с открытым кодом

Национальный аэрокосмический университет им. Н.Е. Жуковского "ХАИ"

Определены особенности автоматизации процесса тестирования уязвимостей web-ресурсов с использованием программного обеспечения с открытым кодом; рассмотрена процедура взаимодействия отдельных модулей тестирования уязвимостей web-ресурсов; предложен механизм взаимодействия отдельных модулей в рамках создания системы автоматизации тестирования уязвимостей web-ресурсов.

Ключевые слова: информационная безопасность, cross-site scripting, XSS уязвимости, web-приложения, автоматизация тестирования уязвимостей web-ресурсов.

Введение

Современные темпы развития информационных технологий настолько высоки, что задачи, связанные с обеспечением информационной безопасности ИТ-ресурсов вообще и Web-ресурсов в частности зачастую остаются решёнными частично, т.е. без проведения детального и всестороннего анализа наличия уязвимостей и возможностей их использования потенциальными злоумышленниками. ИТ-решения на базе открытого кода, активно развивающиеся последние несколько лет, к сожалению, также подвержены подобной тенденции, однако, в отличие от проприетарного ПО, благодаря своим сообществам имеют более высокие шансы раннего обнаружения наличия уязвимостей, его опубликования и устранения путём применения заплат [1-4].

Постановка задачи

Актуальность задачи по автоматизации процесса тестирования уязвимостей web-ресурсов с использованием ПО с открытым кодом заключается в возможности обеспечения повышения эффективности проведения комплекса мероприятий по выявлению брешей в системе информационной безопасности web-ресурсов. Эффективность в данном случае понимается как отношение уменьшения суммарного времени на проведения операций тестирования уязвимостей после внедрения автоматизации к суммарному количеству затраченного времени до автоматизации.

Для организации автоматизированного управления процесса тестирования уязвимостей web-ресурсов ПО должно предоставлять некоторое внешнее API (Application Programming Interface). Важно чтобы API позволяло интерактивно управлять инструментальным средством. Так, например, Vega полностью ориентирована на взаимодействие через графический интерфейс пользователя, но предоставляет API для написания модулей на JavaScript. Тем не менее это не позволяет интерактивно управлять работой программы, что делает инструмент непригодным для управления из других процессов или конфигурации командной строкой.

Большинство инструментальных средств возможно запускать и конфигурировать с помощью командной строки, что делает удобным задание нужных параметров конфигурации, их подстановку и запуск на исполнение.

Инструменты, которые не поддерживают задания конфигурации, обычно хранят ее в конфигурационных файлах, которые тоже можно генерировать динамически.

Исследование и анализ особенностей автоматизации процесса тестирования уязвимостей web-ресурсов с использованием программного обеспечения с открытым кодом ОС Kali Linux [5-8].

Результаты

В процессе создания дополнений к существующим программным средствам довольно часто возникает проблема дефицита знаний о структуре проекта, к которому нужно создать дополнение (модуль, компонент расширения функциональности и т.п.). Не каждый проект имеет документацию, из которой можно узнать его структуру. Поэтому, прежде чем начать разработку какого-либо модуля, разработчику приходится тратить время на изучение структуры. Исследование некоторого устройства или программы с целью понять принцип его работы или воспроизвести структуру называется реверс-инжинирингом (обратной разработкой).

Чтобы построить систему управления набором сканеров веб-ресурсов, нужен механизм для запуска программ на выполнение с передачей им параметров конфигурации. Рассмотрим общие этапы жизненного цикла выполнения работы каждым из инструментов.

Перед запуском инструмента нужно сформировать для него набор параметров. Среди таких параметров могут быть специфические для конкретного инструмента параметры, и общие. Примером общих параметров является адрес ресурса тестируемого, максимальная глубина поиска, и тому подобное. Если инструмент не может быть сконфигурирован параметрами запуска или через конфигурационный файл, его применение в автоматическом режиме не представляется возможным.

Запуск инструментов можно делать последовательно или параллельно. На самом деле, веб-сканеры - достаточно требовательны к системным ресурсам программы, поэтому их параллельное выполнение не дает значительного уменьшения времени выполнения сканирования, но зато значительно усложняет алгоритмы отслеживания работы процессов и управления ими. Поэтому было решено выполнять сканирование набором инструментов последовательно.

После запуска инструмента и начала его работы, устанавливается PID его процесса и начинается наблюдение за выполнением. По окончании выполнения, процесс завершается, собираются итоги его работы, и запускается следующий инструмент из очереди. Во время работы следующего инструмента параллельно может проходить анализ результатов работы предыдущего.

Каждый инструмент имеет свой формат изображения информации. Обычно это текстовые сообщения, выводимые в поток стандартного вывода для просмотра пользователем. Такие сообщения удобно перенаправить в файл для последующего синтаксического и семантического анализа, учета и на финальном этапе, преобразования в единый формат.

После того как последний инструмент завершил свою работу, собираются результаты работы, анализируются и добавляются к учету. Тогда основываясь на всех собранных данных учета, рассчитывается общий итог по найденным уязвимостям, их достоверность и примерный уровень опасности. Общий итог трансформируется в XML-документ и вместе с отдельными отчетами каждого из

инструментов упаковывается в архив. Возможна реализация оповещения пользователя по электронной почте о завершении работы. Такое оповещение актуально из-за большого количества времени что требует сканирование - в зависимости от параметров и размеров веб-ресурса, анализ веб-ресурса некоторыми инструментами может достигать нескольких дней.

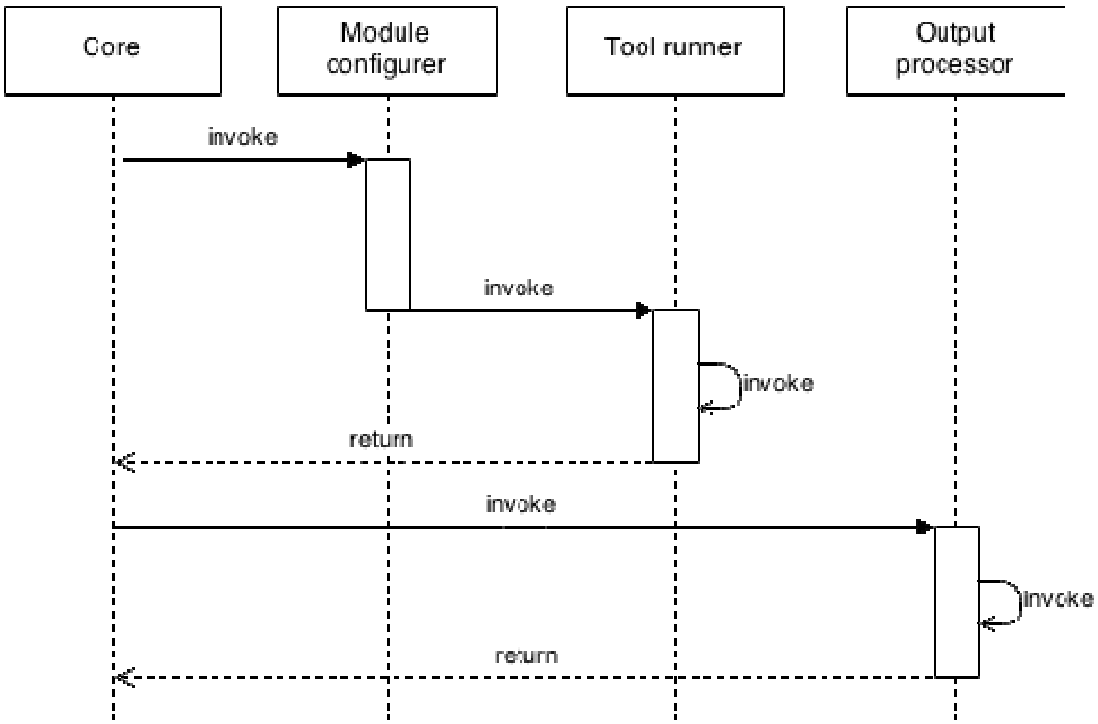

Рис. Диаграмма последовательности работы модулей одного инструмента

Учитывая потребность работы с множеством различных инструментов, что конфигурируются различными способами, имеют разные параметры и свойства, каждый из них формирует вывод в уникальном для инструмента формате, а также для возможности расширения системы автоматизации и легкого включения поддержки новых инструментов, было принято решение строить систему на модульной основе. Ядро с несколькими встроенными модулями возможно дополнять четырьмя видами модулей: модуль конфигурации, модуль выполнения, модуль синтаксического анализа результатов работы, и модуль учета (расчетов).

Модуль конфигурации получает параметры из мастер-конфигурации - файла, который задает общие настройки для всех инструментов, а затем объединяет их с параметрами для конкретного модуля, отдавая предпочтение локальной конфигурации. Модуль конфигурации инкапсулирует логику преобразования общего формата настроек в специфический для инструмента, для которого этот модуль предназначен.

Модуль выполнения стартует инструмент, учитывая подготовленные модулем конфигурации параметры, и следит за жизненным циклом инструмента.

После завершения работы инструмента, модуль синтаксического анализа анализирует отчет и превращает его в единый внутренний формат доменной модели. Этот модуль инкапсулирует нюансы представления информации конкретным инструментом и позволяет другим модулям получать готовые данные в удобном для них формате.

Модуль учета проводит финальные расчеты всех полученных отчетов, отфильтровывает незначительные данные и акцентирует на проблемах, предоставляет статистические данные.

В качестве модели для проверки работы системы выбрана реализация тестового веб-ресурса, который умышленно игнорирует рекомендации OWASP XSS Prevention Cheat Sheet.

Среди рекомендаций необходимо указать список "слотов" HTML-документа, в которые разрешается вставлять экранированные пользовательские данные. К ним относятся: тела HTML-элементов, значения атрибутов HTML-элементов, строковые литералы JavaScript-кода (за некоторыми исключениями), JSON, значения свойств CSS-селекторов (за некоторыми исключениями) и значения параметров URL. Поэтому для проверки работы системы построена испытательная среда, содержащая нарушения правил, описанных в рекомендациях OWASP XSS Prevention Cheat Sheet с различными вариантами и комбинациями других уязвимостей, таких как отсутствие флагов HTTPOnly для cookies и не реализована политика безопасности контента.

На основании результатов исследования и анализа особенностей автоматизации процесса тестирования уязвимостей web-ресурсов с использованием программного обеспечения с открытым кодом было разработано инструментальное средство, позволяющее автоматизировать совместное использование различного ПО, используемого для тестирования уязвимостей.

Выводы

Таким образом, были получены следующие результаты:

- определены особенности автоматизации процесса тестирования уязвимостей web-ресурсов с использованием программного обеспечения с открытым кодом;
- рассмотрена процедура взаимодействия отдельных модулей тестирования уязвимостей web-ресурсов;
- предложен механизм взаимодействия отдельных модулей в рамках создания системы автоматизации тестирования уязвимостей web-ресурсов.

К направлениям дальнейшего исследования можно отнести проведение эксперимента с помощью создаваемого ПО для автоматизации процесса тестирования уязвимостей веб-ресурсов и сравнение результатов с выходными данными по использованию отдельных инструментальных средств пентестинга.

Список литературы

1. https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
2. http://www.pentest-standard.org/index.php/Vulnerability_Analysis
3. http://en.wikipedia.org/wiki/ISO/IEC_27001:2013
4. Offensive Security Ltd. (2012). Introduction - Metasploit Unleashed. - http://www.offensive-security.com/metasploit-unleashed/Introduction
5. https://www.kali.org/
6. OWASP, XSS (Cross Site Scripting) Prevention Cheat Sheet. - https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet
7. Offensive Security Ltd. (2012). Introduction - Metasploit Unleashed. - http://www.offensive-security.com/metasploit-unleashed/Introduction

8. MSDN (2013) - Шаблон разработки Observer - https://msdn.microsoft.com/ru-ru/library/ee850490.aspx

Поступила в редакцию 02.02.2015

Особливості автоматизації процесу тестування уразливостей web-ресурсів з використанням ПЗ з відкритим кодом

Визначено особливості автоматизації процесу тестування уразливостей web-ресурсів з використанням програмного забезпечення з відкритим кодом; розглянута процедура взаємодії окремих модулів тестування уразливостей web-ресурсів; запропоновано механізм взаємодії окремих модулів в рамках створення системи автоматизації тестування уразливостей web-ресурсів.

Ключові слова: інформаційна безпека, cross-site scripting, XSS уразливості, web-додатки, автоматизація тестування уразливостей web-ресурсів.

Automation of pentesting process of web-resources by means of open source software

The features of automation of pentesting process of web-resources by means of open source software are described; a procedure for interaction of the individual modules vulnerability testing web-resources is shown; the mechanism of the interaction of individual modules as part of a web-resources pentesting automation system is proposed.

Keywords: information security, cross-site scripting, XSS vulnerability, web-application, web-resources pentesting automation system.