BlackBox-тестирование

Омар Ганиев
SpbCTF
Санкт-Петербург

Первый слайд

- Beched (@ahack_ru)

План

- Теория кратко
  - Для чего нужен чёрный ящик
  - Что нужно для чёрного ящика
- Детали и практика
  - Быстрый поверхностный анализ сайта
  - Фаззинг и фингерпринтинг интерфейсов
  - Анализ логики и «угадывание» уявимостей
- Актуальные исследования
  - Ограничения автоматических инструментов
  - Возможные решения

1. Теория вкратце

- Зачем бизнесу такая услуга как тестирование чёрным ящиком
- Зачем учиться ломать именно блекбоксом
- Какие особенные или нетривиальные навыки для этого потребуются

Причины для чёрного ящика

- Недоверие, NDA или лицензионные ограничения для выдачи исходных текстов
- Дороговизна анализа большого сложного проекта в исходниках (автоматический статанализ тоже неэффективен)
- Моделирование действий реальных нарушителей (Red Team, BugBounty)
- Сканеры плохо справляются, поэтому браться за дело приходится людям

С точки зрения пентестера

- У blackbox-тестирования ниже порог входа, каждый может взять burp suite и начать тыкать кавычки
- Но сложные вещи намного проще понять, имея исходники, поэтому blackbox-тестирование требует определённых навыков, знаний, методик и инструментов
- Самое интересное – идеи и методики, порой позволяющие реверсить нестандартную логику

Навыки

- Хакерская догадка, угадайка, интуиция, рандом и т.д. – умение строить «неожиданные» правильные догадки о server-side-логике приложения
- Умение пошагово анализировать поведение приложения, выявляя закономерности и алгоритм работы

Знания

- Языки программирования, их особенности и специфичные уязвимости
- Архитектура и парадигмы программирования веб-приложений
- Веб-серверы и их конфигурация, Linux
- Архитектура распределённых и высоконагруженных систем

Методики

- Автоматическое сканирование различными сканерами
- Фингерпринтинг используемого стека технологий, сторонних фреймворков, CMS и библиотек
- Ручной анализ логики работы и семантики приложения
- Фаззинг точек входа для «реверс-инжиниринга» приложения

Инструменты

- Большой пласт задач решается нативно или через расширения в Burp Suite
- Сканеры: Acunetix WVS, w3af, Nessus, arachni, AppScan, Netsparker, OWASP ZAP, Burp Suite, …
- Различные утилиты: subbrute, dirb, расширения Burp Suite, libpywebhack
- Фингерпринтинг, скан конкретных CMS: whatweb, wpscan, joomscan, droopescan
- GitHub, Python, https://github.com/enaqx/awesome-pentest

2. Детали и практика

- Приблизительно разберём по шагам мыслительный и практический процесс blackbox-тестирования
- Не теряя времени, сразу же проделаем некоторые вещи на практике
- Адрес сервера: http://spbctf.ahack.ru/ (95.213.200.83)
- Софт: https://yadi.sk/d/qYTh6t_h33w3Zd
- На слайдах текстовые комментарии
- Автор таска phpsocute -- @d90andrew для phd hackquest

Идентификация

- Для анализа веб-приложения нужно определить scope или область исследования
- Какие сетевые узлы и доменные имена обслуживают сайт или связаны с ним
- Для каждого узла просканировать порты, обнаружить имеющиеся веб-серверы
- Для каждого веб-сервера найти виртуальные хосты, перебирать файловую структуру

Hands-on

- Для разогрева решили простой таск двумя способами: https://yadi.sk/d/QiN-q0_l33w4iH
- Умеем перебирать файловую структуру и искать интересные файлы
- Умеем обелять чёрный ящик при ошибках деплоя: скачивание git-репозитория, поиск временных файлов nano, vim и т.д.
- Умеем перебирать входные HTTP-параметры чёрного ящика и делать прочие мелкие плюшки

Разведка

- Необходимо ответить на вопросы:
  - Какой стек технологий (балансировщики, application-серверы, фреймворки, CMS, библиотеки, языки программирования) использует сайт?
  - Какие функции предоставляет этот сайт?
  - Какие компоненты сайты являются самописными, а какие сторонними

Hands-on

- По PTR-записи обнаружили скрытый виртхост с таском, определили веб-сервер, язык программирования, CMS
- Сигнатуры CMS оказались фейковые
- Сайт самописный, обнаружен только скрипт с phpinfo
- Для всех .html-файлов ответ 500, кроме index.html, причем он тоже обрабатывается PHP
- Как это работает?

Осмысление

- Далее задаём следующие вопросы:
  - Как именно разработчик мог реализовать ту или иную функцию сайта?
  - Какие ошибки в коде и в каких случаях он мог допустить?
  - Какие могли быть ошибки в окружении при деплое и конфигурации?

Hands-on

- Давайте осмыслять. При ошибке 500 страница не дорисована, значит, возникла fatal error
- Значит, текущий REQUEST_URI каким-то образом обрабатывается php-скриптом, если заканчивается на .html
- При этом обнаружено только одно валидное имя – index
- Понятно, что сделано это при помощи mod_rewrite с правилом вида ^(.+).html$ handler.php?file=$1

Анализ

- В итоге получаем дерево возможных вариантов реализации каждой интересующей нас функции
- Путём фаззинга проверяем каждый вариант, проверяя какой из них рабочий
- Для каждой верной цепочки догадок проверяем наличие уязвимости

Hands-on

- Может, есть белый список имён страниц (“index”) и содержимое для каждой
- Может, идёт обращение к СУБД, где прописаны роуты. Тогда возможна инъекция. Но кавычкопихательство ничего не даёт
- А что если скрипт просто читает страницу из файла, имя которого берёт из параметра?
- Тогда мы контролируем это имя файла, не зная ни имя скрипта, ни имя параметра
- Для того, чтобы сделать обход директории в REQUEST_URI, нужен двойной url-encode: %252f..%252f
- /..%252f..%252f..%252f..%252f..%252fetc%252fpasswd.html

Hands-on

- Ну ни хрена ж себе, это действительно работает!
- Это угадайка? Нет, мы поняли по выхлопу dirb архитектуру сайта и перебрали возможные реализации кода и соответствующие ошибки

Hands-on

- Альтернативный путь – давайте тыкать. У нас есть phpinfo.
- Apache по умолчанию вызывает php-скрипт, даже если при обращении продолжить путь script.php*/asdqwe…/test.html*
- То, что курсивом – это PATH_INFO. А есть ещё PATH_TRANSLATED – серверный путь к текущему файлу
- Phpinfo отображает суперглобальные массивы, так что попробуем сделать хрень: /pi.php/test.html

Hands-on

- Видим: PATH_TRANSLATED redirect:**/t3mp473l04d3r.php**
- Мы правы, и .html-файлы обрабатываются php-скриптом t3mp473l04d3r.php
- Перебираем параметры, чтобы дальше фаззить их без ограничений RFC на REQUEST_URI
- Поскольку вся эта логика может быть связана с ФС, попробуем пейлоады /../../../etc/passwd и /etc/passwd

Hands-on

- Мой кривой скрипт не справляется с пейлоадом с “/../” из-за зависания сервера
- Но по “/etc/passwd” сразу обнаруживаются 2 параметра: template и dir
- Действительно: /t3mp473l04d3r.php?dir=/etc/passwd

Анализ

- Цепочка догадок – это гипотезы о работе интерфейса на разных уровнях, т.е. какой путь проделывает пользовательский запрос (веб-сервер, приложение, СУБД, ФС и т.д.)
- Одного интерфейса может быть недостаточно для понимания структуры приложения и обнаружения уязвимости
- Пересечение догадок и debug-информация дают более точную картину

Hands-on

- Мы прочитали passwd, это «читалка»? Или PHP-LFI?
- Вспоминаем, что у нас есть phpinfo, а через него можно узнать имя временного файла для проведения LFI
- Пробуем стандартным скриптом – плейнтекст подключается, но PHP-код падает с 500

Hands-on

- Очевидно, это не include и не readfile, но это динамическое исполнение кода
- Какой ещё код может исполняться в PHP, но отличаться по синтаксису?
- Шаблоны! Например, Smarty
- Пробуем выполнить PHP-через smarty-выражения и получаем RCE

Вывод

- Как видно из решённой задачи, нам понадобились и инструменты, и методики, и навыки и знания: https://yadi.sk/d/BHn852Zs33w4kB
- Часто, чтоб найти ошибку, нужно мыслить как программист, который её совершает
- А для этого нужно знать технологии разработки и языки программирования

Пример

- Простой и рутинный пример из практики, очень похожий на CTF-таск Бума с того же phd hackquest
- Есть форма ввода строки (имени), сервер выдаёт в ответ адрес картинки, на которой это строка написана
- Как действовать?

Пример

В параметре	На картинке
asdqwe'"\	asdqwe'"
asd\nqwe	asd qwe
asd!@#$%^&*()_+-=./,|	asd!@#$&*()_+-=./,|
%%	%
%x	72 Undefined
%d	./assets
%r	DirectClass sRGB Matte
%[version]	ImageMagick 6.7.7-10 2014-03-06 Q16

Пример

- В таблице показан последовательный перебор значений для выявления управляющих символов и определения синтаксиса
- По поведению % стало ясно, что это форматная строка
- По 2-3 значениям удалось определить, что это формат imagemagick (что можно было угадать и так)

Бонус

- Мы разбирали CTF-таски, но речь шла о реальных приложениях и подходах в пентесте
- В соревнованиях есть свои особенности
- В CTF приложения обычно маленькие, и почти всё сделано не зря, а для решения задачи
- Заведомо зная, что в сервис заложена уязвимость, можно по имеющимся интерфейсам понять, какая она, и как надо решать
- В CTF не часто бывают побочные уязвимости, которые не дают решения

Бонус

- В уже разобранной задаче увидели phpinfo – значит, будет LFI
- Видим отправку личных сообщений «администратору» – будет XSS с ботом
- Есть админка – надо в неё попасть
- Есть недоступный флаг (например, как товар в магазине) – надо поднять привилегии или применить логический баг
- Есть загрузка файлов – надо её использовать
- Флаг предположительно в ФС – нужна читалка или RCE

Бонус

- Также полезно узнавать автора таска и смотреть его ресерчи и релизы, недавно обнаруженные им или популярные баги и т.д.
- В A&D-цтфах белый ящик, но особенность в том, что надо как можно раньше начать собирать флаги, часто чёрным ящиком нужный баг можно отыскать быстрее, чем если читать исходники

Бонус

- BugBounty – тоже соревнование, но в нём очень большой скоуп, нет заложенных багов, и баг можно сдать только один раз
- Есть множество презентаций на тему методологии в BB: lmgtfy
- Вкратце: нужно максимально автоматизировать все проверки и постоянно повторять стандартные сканеры, покрывая новые хосты и скрипты
- Также нужно находить нестандартные уязвимости, под которые тоже можно писать чекеры и гонять их автоматом

3. Актуальные исследования

- Принципиально методология blackbox-тестирования давно описана, например, в OWASP
- Наиболее сложная и интересная тема – эффективный автоматический blackbox-скан
- Фактически его полноценное создание требует создания искусственного интеллекта
- На текущий момент на рынке нет state-of-art-сканера, лидеры рынка существенно не дотягивают до человека-пентестера

Ссылки

- Про «теорию»:
  - http://blog.portswigger.net/2016/11/backslash-powered-scanning-hunting.html
  - http://artsploit.com/AdvancedWebAppFuzzing.pptx
  - http://www.slideshare.net/beched/find-maximum-bugs-in-limited-time
  - http://www.slideshare.net/beched/data-mining-for-nmap-acceleration
  - http://2015.zeronights.org/assets/files/04-Novikov.pdf

Ссылки

- Про «практику»:
  - http://www.slideshare.net/bugcrowd/bug-bounty-hunter-methodology-nullcon-2016
  - https://www.owasp.org/index.php/Appendix_A:_Testing_Tools
  - https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents
  - https://habrahabr.ru/company/mailru/blog/202554/
  - http://www.slideshare.net/beched/slides-33757106

Задачи для сканера

- Определять семантику приложения, смысл интерфейсов
- Тестировать не только инъекции кода, но и логические ошибки
- Отсеивать повторные проверки, эффективно делать гипотезы и проводить фаззинг в соответствии с ними
- Уметь вычленять элементы страницы, отличать артефакты уязвимостей от иного динамического содержимого

Задачи для сканера

- Одним словом, сканер должен адаптивно эмулировать сознательные действия человека, а не перебирать сигнатуры
- Для этого его в идеале нужно научить естественному языку, программированию, сетевым технологиям и т.д.

Что можно сделать

- Для решения недостатков сканеров можно сделать разные шаги:
  - Провести исследование и получить методику, делающую сканирование эффективнее
  - На основе этой методики создать свой кривой сканер для себя, который стыдно показывать (мой путь)
  - Писать плагины под существующие популярные инструменты: Burp Suite, w3af, …

Спасибо за внимание!

admin@ahack.ru

beched@incsecurity.ru