Лекция №9 Тестирование безопасности

### **Введение**

[**Тестирование безопасности**](https://dist.belstu.by/mod/assign/view.php?id=57440) - это метод тестирования, позволяющий определить, защищена ли информационная система данными и поддерживает ли она функциональность по назначению. [Тестирование безопасности](https://dist.belstu.by/mod/assign/view.php?id=57440) не гарантирует полную безопасность системы, но важно включить [тестирование безопасности](https://dist.belstu.by/mod/assign/view.php?id=57440) как часть процесса тестирования.

[Тестирование безопасности](https://dist.belstu.by/mod/assign/view.php?id=57440) включает следующие шесть мер для обеспечения защищенной среды:

* Конфиденциальность. Он защищает от раскрытия информации непреднамеренным получателям.
* Целостность - позволяет передавать точную и правильную требуемую информацию от отправителей к предназначенным приемникам.
* Аутентификация - проверяет и подтверждает личность пользователя.
* Авторизация - указывает права доступа к пользователям и ресурсам.
* Доступность - это обеспечивает готовность информации о требованиях.
* Неотказание - оно гарантирует, что от отправителя или получателя не будет отказано в отправке или получении сообщения.

**Пример**

Определение недостатка безопасности в веб-приложении требует сложных шагов и творческого мышления. Иногда простой тест может подвергать самый серьезный риск безопасности. Можно попробовать самый базовый тест для любого веб-приложения:

* Войти в веб-приложение, используя действительные учетные данные.
* Выйти из веб-приложения.
* Нажать кнопку BACK в браузере.
* Убедиться, что пользователя попросили снова войти в систему или снова вернуться на страницу входа.

### **Процесс тестирования безопасности**

[Тестирование безопасности](https://dist.belstu.by/mod/assign/view.php?id=57440) можно рассматривать как контролируемую атаку на систему, которая реалистично раскрывает недостатки безопасности. Его цель - оценить текущий статус ИТ-системы. Он также известен как тест на проникновение или более популярным, как этический взлом.

Тест на проникновение проводится поэтапно. Надлежащая документация должна выполняться на каждом этапе, чтобы все этапы, необходимые для воспроизведения атаки, были доступны с готовностью. Документация также служит основой для получения подробного отчета, который клиенты получают в конце теста на проникновение.

#### **Тест проницаемости - рабочий процесс**

Тест на проникновение включает четыре основные фазы:

1. Сбор информации (footprinting)
2. Сканирование
3. Перечисление
4. Эксплуатация

#### **Footprinting**

**Footprinting** - это процесс сбора плана конкретной системы или сети и устройств, которые подключены к рассматриваемой сети. Это первый шаг, который использует тестировщик проникновения для оценки безопасности веб-приложения.

**Footprinting - Шаги**

* Сбор информации
* Определение диапазона сети
* Идентификация активных машин
* Идентификация открытых портов и точек доступа
* Отпечатки пальцев OS
* Услуги по отпечатку пальцев
* Сопоставление сети

**Инструменты, используемые при слежении**

Ниже приведен общий набор инструментов, используемых для footprinting:

* Кто
* SmartWhois
* NSlookup
* Сэм Спейд

**Другие методы, используемые при слежении**

Footprinting может также включать сбор информации, такой как:

* Контактные лица компаний, адреса электронной почты и номера телефонов
* Сделки с компанией и другие вовлеченные стороны
* Новости о слияниях и поглощениях
* Ссылки на другие сайты, связанные с компанией
* Политика конфиденциальности компании

#### **Сканирование**

**Сканирование** - это второй шаг, который выполняется после footprinting. Он включает в себя сканирование открытых портов, отпечатки пальцев операционной системы и обнаружение служб на портах. Конечной целью сканирования является поиск открытых портов через внешнее или внутреннее сканирование сети, пинговые машины, определение сетевых диапазонов и сканирование портов на отдельных системах.

**Инструменты, используемые в сканировании**

Ниже приведен общий набор инструментов / ресурсов, используемых в Scanning:

* NMap
* Пинг
* Трассировка
* SUPERSCAN
* Netcat
* NeoTrace

#### **Перечисление**

**Перечисление**- следующий шаг после сканирования. Цель перечисления - получить полную картину цели. На этом этапе тестировщик проникновения пытается определить действительные учетные записи пользователей или плохо защищенные общие ресурсы, используя активные подключения к системам.

**Методы, используемые в перечислении**

Ниже приведен общий набор процедур, используемых в перечислении:

* Идентификация уязвимых учетных записей пользователей
* Получение информации в Active Directory
* Использование snmputil для перечисления протокола Simple Network Management
* Использование DNS-запросов Windows
* Создание нулевых сеансов и соединений

#### **Эксплуатация**

**Эксплуатация**- это последний этап, когда тестировщик безопасности активно использует уязвимости безопасности, присутствующие в рассматриваемой системе. Как только атака будет успешной, можно проникнуть в другие системы в домене, потому что тестировщики проникновения затем имеют доступ к более потенциальным целям, которые раньше не были доступны.

**Методы, используемые при эксплуатации**

Типы эксплуатации подразделяются на три категории:

* Атака против WEB-СЕРВЕРОВ
  + SQL-инъекция
  + Межсайтовый скриптинг
  + Ввод кода
  + Захват сеанса
  + Обход каталога
* Атака на NETWORKS
  + Человек в средней атаке
  + Подделка
  + Обход брандмауэра
  + WLAN
  + Отравление ARP
* Атака на услуги
  + Переполнение буфера
  + Форматировать строки
  + ДОС
  + Ошибки аутентификации

Начало формы

Вредоносное ПО

Конец формы

### **Вредоносное ПО**

**Вредоносная программа (вредоносное ПО)** - это любое программное обеспечение, которое частично контролирует систему для злоумышленника / создателя вредоносного ПО.

#### **Malwares**

Ниже перечислены различные виды вредоносного ПО:

**Вирус** - это программа, которая сама создает копии и вставляет эти копии в другие компьютерные программы, файлы данных или в загрузочный сектор жесткого диска. При успешной репликации вирусы вызывают вредную активность на зараженных компьютерах, например, кражу места на жестком диске или процессорного времени.

**Червь** - это тип вредоносного ПО, который оставляет свою копию в памяти каждого компьютера на своем пути.

**Trojan** - не самовоспроизводящийся тип вредоносного ПО, содержащий вредоносный код, который после выполнения приводит к потере или краже данных или возможному системному вреду.

**Adware** - рекламное ПО, также известное как бесплатное программное обеспечение или программное обеспечение для катания на потолке, представляет собой бесплатное программное обеспечение для компьютеров, которое содержит коммерческую рекламу игр, настольных панелей инструментов и утилит. Это веб-приложение, и оно собирает данные веб-браузера для таргетинга рекламных объявлений, особенно всплывающих окон.

**Spyware**- программное обеспечение для инфильтрации, которое анонимно контролирует пользователей, что позволяет хакеру получать конфиденциальную информацию с компьютера пользователя. Spyware использует уязвимости пользователей и приложений, которые нередко привязаны к бесплатной загрузке онлайн-программного обеспечения или к ссылкам, которые пользователи кликают.

**Rootkit**- это программное обеспечение, используемое хакером для доступа к уровню доступа администратора к компьютеру / сети, который устанавливается через украденный пароль или путем использования уязвимости системы без знаний жертвы.

#### **Программное обеспечение для защиты от вредоносных программ**

Следующее программное обеспечение поможет удалить из системы malwares -

* Основы безопасности Microsoft
* Защитник Microsoft Windows
* AVG Internet Security
* Spybot - поиск и уничтожение
* Avast! Домашнее издание для личного пользования
* Panda Internet Security
* MacScan для Mac OS и Mac OS X

### **Основы протокола HTTPS**

**HTTPS**(протокол передачи гипертекста через Secure Socket Layer) или **HTTP over SSL** - это веб-протокол, разработанный Netscape. Это не протокол, но это результат преобразования HTTP поверх SSL / TLS (Secure Socket Layer / Transport Layer Security).

Упрощенно говоря, HTTPS = HTTP + SSL

#### **Когда требуется HTTPS?**

Когда мы просматриваем веб-страницы, мы обычно отправляем и получаем информацию по протоколу HTTP. Поэтому это приводит к тому, что кто-либо подслушивает разговор между нашим компьютером и веб-сервером. Много раз нам нужно обмениваться конфиденциальной информацией, которую необходимо обеспечить, и предотвращать несанкционированный доступ.

Протокол Https, используемый в следующих сценариях:

* Банковские сайты
* Платежный шлюз
* Торговые сайты
* Все страницы входа
* Приложения электронной почты

#### **Основная работа HTTPS**

* Для сервера в протоколе HTTPS требуется открытый ключ и подписанные сертификаты.
* Запросы клиентов для страницы https: //
* При использовании https-соединения сервер отвечает на первоначальное соединение, предлагая список методов шифрования, поддерживаемых веб-сервером.
* В ответ клиент выбирает метод подключения, а клиентские и серверные обменяет сертификаты для аутентификации своих идентификаторов.
* После этого оба веб-сервера и клиента обмениваются зашифрованной информацией после того, как обе используют один и тот же ключ, и соединение закрыто.
* Для хостинга https-соединений сервер должен иметь сертификат открытого ключа, который включает ключевую информацию с проверкой личности владельца ключа.
* Почти все сертификаты проверяются третьей стороной, чтобы клиенты были уверены, что ключ всегда безопасен.

Начало формы

Кодирование и декодирование

Конец формы

### **Кодирование и декодирование**

**Кодирование**- это процесс размещения последовательности символов, таких как буквы, цифры и другие специальные символы, в специализированный формат для эффективной передачи.

**Декодирование** - это процесс преобразования кодированного формата обратно в исходную последовательность символов. Он полностью отличается от шифрования.

Кодирование и декодирование используются при передаче и хранении данных. Кодирование НЕ должно использоваться для транспортировки конфиденциальной информации.

**Кодирование URL**

URL-адреса можно отправлять только через Интернет с использованием набора символов ASCII, и есть экземпляры, когда URL содержит специальные символы, кроме символов ASCII, его необходимо закодировать. URL-адреса не содержат пробелов и заменяются знаком плюс (+) или% 20.

**Кодирование ASCII**

Браузер (клиентская сторона) будет кодировать ввод в соответствии с набором символов, используемым на веб-странице, а набор символов по умолчанию в HTML5 - это UTF-8.

### **Криптография**

**Криптография**- это наука для шифрования и дешифрования данных, которая позволяет пользователям хранить конфиденциальную информацию или передавать ее через небезопасные сети, чтобы ее можно было читать только предполагаемого получателя.

Данные, которые могут быть прочитаны и поняты без каких-либо специальных мер, называются открытым текстом , а метод маскировки открытого текста для того, чтобы скрыть его вещество, называется шифрованием .

Зашифрованный открытый текст известен как шифрованный текст, и процесс возврата зашифрованных данных обратно в обычный текст называется расшифровкой .

Наука о анализе и разрыве безопасной связи известна как криптоанализ.

Криптография может быть сильной или слабой, а сила измеряется временем и ресурсами, которые потребуются для восстановления фактического открытого текста.

Следовательно, для расшифровки сильных зашифрованных сообщений требуется соответствующий инструмент декодирования.

Существуют некоторые криптографические методы, с помощью которых даже миллиард компьютеров, выполняющих миллиард проверок в секунду, невозможно расшифровать текст.

Поскольку вычислительная мощность растет с каждым днем, нужно сделать алгоритмы шифрования очень сильными, чтобы защитить данные и критическую информацию от злоумышленников.

#### **Как работает шифрование?**

Криптографический алгоритм работает в сочетании с ключом (может быть слово, число или фраза) для шифрования открытого текста, а один и тот же открытый текст шифрует различный шифрованный текст с помощью разных ключей.

Следовательно, зашифрованные данные полностью зависят от нескольких параметров, таких как сила криптографического алгоритма и секретность ключа.

#### **Криптографические методы**

**Симметричное шифрование**. Обычная криптография, также известная как обычное шифрование, - это метод, в котором для шифрования и дешифрования используется только один ключ. Например, DES, Triple DES алгоритмы, MARS от IBM, RC2, RC4, RC5, RC6.

**Асимметричное шифрование**. Это криптография с открытым ключом, которая использует пару ключей для шифрования: открытый ключ для шифрования данных и закрытый ключ для дешифрования. Публичный ключ публикуется людям, сохраняя секретный ключ. Например, RSA, Алгоритм цифровой подписи (DSA), Elgamal.

**Хеширование** - это шифрование ONE-WAY, которое создает скремблированный вывод, который не может быть отменен или, по крайней мере, не может быть легко изменен. Например, алгоритм MD5. Он используется для создания цифровых сертификатов, цифровых подписей, хранения паролей, проверки связи и т. д.

### **Тестирование на проникновения**

#### **Инъекция**

Инъекционная техника состоит из ввода SQL-запроса или команды с использованием полей ввода приложения.

Успешная SQL-инъекция может считывать, изменять конфиденциальные данные из базы данных и также удалять данные из базы данных. Он также позволяет хакеру выполнять административные операции с базой данных, такие как завершение работы СУБД / удаляющихся баз данных.

**Предотвращение внедрения SQL**

Существует множество способов предотвратить SQL-инъекцию. Когда разработчики пишут код, они должны обеспечить, чтобы они обрабатывали специальные символы соответственно. В OWASP (<https://www.owasp.org/>) есть чит-листы / методы профилактики, которые, безусловно, являются руководством для разработчиков.

#### **Межсайтовый скриптинг**

Cross-Site Scripting (XSS) происходит всякий раз, когда приложение берет ненадежные данные и отправляет их клиенту (браузеру) без проверки. Это позволяет злоумышленникам выполнять вредоносные скрипты в браузере жертвы, что может привести к захвату пользовательских сеансов, защите веб-сайтов или перенаправлению пользователя на вредоносные сайты.

**Типы XSS**

**Хранимый XSS,** также известный как постоянный XSS, возникает, когда пользовательский ввод хранится на целевом сервере, таком как поле / комментарий форума / комментария базы данных и т. д. Затем жертва может извлекать сохраненные данные из веб-приложения.

**Отраженный XSS**, также известный как непостоянный XSS, возникает, когда пользовательский ввод немедленно возвращается веб-приложением в результате сообщения об ошибке / результате поиска или ввода, предоставленного пользователем как часть запроса, и без постоянного хранения предоставленных пользователем данных.

**Основанный на DOM XSS.** XSS на основе DOM является формой XSS, когда источник данных находится в DOM, приемник также находится в DOM, и поток данных никогда не покидает браузер.

#### **Конфигурация**

Конфигурация безопасности возникает, когда параметры безопасности определены, реализованы и сохранены как значения по умолчанию. Для обеспечения безопасности требуется безопасная конфигурация, определенная и развернутая для приложения, веб-сервера, сервера базы данных и платформы. Не менее важно обновить программное обеспечение.

Некоторые классические примеры ошибочной настройки безопасности приведены как:

* Если список каталогов не отключен на сервере, и если злоумышленник обнаруживает то же самое, злоумышленник может просто перечислить каталоги, чтобы найти любой файл и выполнить его. Также возможно получить фактическую базу кода, которая содержит весь ваш собственный код, а затем найти серьезные недостатки в приложении.
* Конфигурация сервера приложений позволяет отслеживать трассировки стека для пользователей, потенциально подвергая уязвимости. Атакующие захватывают ту дополнительную информацию, которую предоставляют сообщения об ошибках, которых достаточно для их проникновения.
* Серверы приложений обычно поставляются с примерами приложений, которые недостаточно хорошо защищены. Если не удалено с сервера производства, это приведет к компрометации вашего сервера.

#### **Открытые данные**

Поскольку онлайн-приложения продолжают изо дня в день затоплять интернет, не все приложения защищены. Многие веб-приложения не обеспечивают надлежащую защиту конфиденциальных пользовательских данных, таких как информация о кредитных картах / учетные данные банковской учетной записи / учетные данные. Хакеры могут в конечном итоге украсть эти слабо защищенные данные для совершения мошенничества с кредитными картами, кражи личных данных или других преступлений.

Некоторые из классических примеров неправильной конфигурации безопасности приведены:

* Сайт просто не использует SSL для всех аутентифицированных страниц. Это позволяет злоумышленнику контролировать сетевой трафик и красть куки-файлы сеанса пользователя, чтобы захватить сеанс пользователей или получить доступ к их личным данным.
* Приложение хранит номера кредитных карт в зашифрованном формате в базе данных. После извлечения они дешифруются, позволяя хакеру выполнить атаку SQL-инъекции, чтобы получить всю конфиденциальную информацию в ясном тексте. Этого можно избежать, зашифровав номера кредитных карт с помощью открытого ключа и разрешив серверным приложениям расшифровать их с помощью закрытого ключа.

#### **Межсайтовая подделка запроса**

Атака CSRF (Cross-Site Request Forgery) заставляет аутентифицированного пользователя (жертву) отправлять поддельный HTTP-запрос, в том числе куки-файл сеанса жертвы, в уязвимое веб-приложение, которое позволяет злоумышленнику заставить браузер жертвы генерировать запрос, чтобы уязвимое приложение воспринимало как законные запросы от жертва.

**Профилактические механизмы**

CSRF можно избежать, создав уникальный токен в скрытом поле, которое будет отправлено в тело HTTP-запроса, а не в URL-адрес, который более подвержен воздействию.

#### **DoS**

**Атака на отказ в обслуживании** (DoS) является попыткой хакеров сделать сетевой ресурс недоступным. Обычно это прерывает хост, временный или неопределенный, который подключен к Интернету. Эти атаки обычно нацелены на службы, размещенные на критически важных веб-серверах, таких как банки, шлюзы для оплаты кредитных карт.