# Аннотация

В статье описаны основные методы оценки уязвимостей. Кроме того, представлена статистическая выборка данных, связанная с уязвимостями веб-приложений. Инфографика и выводы по данной теме.

# Ключевые слова

Уязвимости, web, OWASP, CVSS, fingerprinting, XSS, CSRF, CSP, SQL-inj, эксперимент, статистика, инфографика.

# Общая информация о веб-уязвимостях

Необходимо понимать, что любые уязвимости, не обязательно веб, необходимо каким-то способом оценивать. Составлять списки самых опасных и наиболее легко эксплуатируемых из них. Именно для этого существуют системы оценки уязвимостей.

Относительно к веб-уязвимостям можно назвать 2 сильные системы оценок – OWASP (Open Web Application Security Project) и CVSS (Common Vulnerability Scoring System). Каждая из них базируется на собственных принципах, но обе они призваны облегчить оценку опасности уязвимостей с помощью изменяемых параметров и их стоимости.

## **Owasp**

Open Web Application Security Project (OWASP) — это открытый проект обеспечения безопасности веб-приложений. Сообщество OWASP включает в себя корпорации, образовательные организации и частных лиц со всего мира. Сообщество работает над созданием статей, учебных пособий, документации, инструментов и технологий, находящихся в свободном доступе.

Сам из себя OAWSP представляет список из наиболее опасных уязвимостей. Так называемый OWASP Top 10. Выглядит он так:

* A1 Внедрение кода
* A2 Некорректная аутентификация и управление сессией
* A3 Межсайтовый скриптинг
* A4 Нарушение контроля доступа
* A5 Небезопасная конфигурация
* A6 Утечка чувствительных данных
* A7 Недостаточная защита от атак
* A8 Подделка межсайтовых запросов
* A9 Использование компонентов с известными уязвимостями
* A10 Незащищенный API

## **CSVV**

Стандарт Common Vulnerability Scoring System был разработан группой экспертов по безопасности National Infrastructure Advisory Council. В эту группу вошли эксперты из различных организаций, таких как CERT/CC, Cisco, DHS/MITRE, eBay, IBM Internet Security Systems, Microsoft, Qualys, Symantec.

В 2005 году состоялась первая публикация стандарта. Основные принципы расчета метрики уязвимостей, изначально заложенные в стандарт, сохранились и по сей день.

Далее стандарт стал поддерживаться рабочей группой Common Vulnerability Scoring System-Special Interest Group (CVSS-SIG) в рамках проекта Forum of Incident Response and Security Teams (FIRST). Членство в группе не накладывает на ее участников каких-либо ограничений по поддержке и распространению стандарта.

CVSS предлагает простой инструментарий для расчета числового показателя по десятибалльной шкале, который позволяет специалистам по безопасности оперативно принимать решение о том, как реагировать на ту или иную уязвимость. Чем выше значение метрики, тем более оперативная реакция требуется.

Примером метрики может служить эта таблица:

|  |  |  |
| --- | --- | --- |
| **Уязвимость** | **Вектор CVSSv3** | **Оценка CVSSv3** |
| [CVE-2014-0160](http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160) | [AV:N/AC:L/PR:N/UI:N/S:U/**C:H**/I:N/A:N](https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N) | 7.5 |
| [CVE-2015-4202](http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-4202) | [AV:N/AC:L/PR:N/UI:N/S:U/**C:L**/I:N/A:N](https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N) | 5.3 |

В ней представлен пример того, насколько гибкой может быть система, использующая вектора атаки, и в данном случае отличие оценки в случае отличия вектора «Метрики воздействия», т.е. оценки степени влияния на конфиденциальность, целостность и доступность атакуемого компонента.

## **Fingerprinting**

**Описание**

**Fingerprinting** – сбор информации о точной версии операционной системы и публичных сервисов.

Именно с данного типа атаки начинается любая серьезная операция по проникновению или уничтожению/краже данных. Злоумышленнику необходимо запастись информацией о вашей системе и работающих в ней приложениях. Для этого может быть использованы любые методы.

Ярким примером будет «Сбор баннеров» - опрос открытых в системе сервисов и возвращаемых ими стандартных приглашений (баннеров).

Так, например, ftp-сервис может позволить выполнение команды SYST, которая выдает версию операционной системы. Заголовок веб-сервера можно получить командой:

**# echo ‘GET / HTTP/1.0 ’ | nc example.com 80 | grep ‘^Server:’**

|  |
| --- |
| **Server: Apache/2.0.52 (Unix)** |

## **Credential/Session Prediction**

**Описание**

Предсказуемое значение идентификатора сессии позволяет перехватывать сессии других пользователей. Подобные атаки выполняются путем предсказания или угадывания уникального идентификатора сессии пользователя.  
  
**Пример**

Многие серверы генерируют идентификаторы сессии, используя алгоритмы собственной разработки. Подобные алгоритмы могут просто увеличивать значение идентификатора для каждого запроса пользователя. Другой распространенный вариант – использование функции от текущего времени или других специфичных для компьютера данных.

## **XSS (Cross-site Scripting)**

**Описание**

Наличие уязвимости Cross-site Scripting позволяет атакующему передать серверу исполняемый код, который будет перенаправлен браузеру пользователя. Этот код обычно создается на языках HTML/JavaScript, но могут быть использованы VBScript, ActiveX, Java, Flash, или другие поддерживаемые браузером технологии.

Переданный код исполняется в контексте безопасности (или зоне безопасности) уязвимого сервера. Используя эти привилегии, код получает возможность читать, модифицировать или передавать важные данные, доступные с помощью браузера.

Существует два типа атак, приводящих к межсайтовому выполнению сценариев: постоянные (сохраненные) и непостоянные (отраженные). Основным отличием между ними является то, что в отраженном варианте передача кода серверу и возврат его клиенту осуществляется в рамках одного HTTP-запроса, а в хранимом – в разных.

Осуществление непостоянной атаки требует, чтобы пользователь перешел по ссылке, сформированной злоумышленником (ссылка может быть передана по email, ICQ и т.д.). В процессе загрузки сайта код, внедренный в URL или заголовки запроса, будет передан клиенту и выполнен в его браузере.

Сохраненная разновидность уязвимости возникает, когда код передается серверу и сохраняется на нем на некоторый промежуток времени. Наиболее популярными целями атак в этом случае являются форумы, почта с Веб-интерфейсом и чаты. Для атаки пользователю не обязательно переходить по ссылке, достаточно посетить уязвимый сайт.

## **SQL-inj**

**Описание**

Эти атаки направлены на Веб-серверы, создающие SQL запросы к серверам СУБД на основе данных, вводимых пользователем.  
Если информация, полученная от клиента, должным образом не верифицируется, атакующий получает возможность модифицировать запрос к SQL-серверу, отправляемый приложением. Запрос будет выполняться с тем же уровнем привилегий, с каким работает компонент приложения, выполняющий запрос (сервер СУБД, Веб-сервер и т.д). В результате злоумышленник может получить полный контроль на сервере СУБД и даже его операционной системой. С точки зрения эксплуатации SQL Injection очень походит на LDAP Injection.

## **CSRF**

**Описание**

Вид атак на посетителей веб-сайтов, использующий недостатки протокола HTTP. Если жертва заходит на сайт, созданный злоумышленником, от её лица тайно отправляется запрос на другой сервер (например, на сервер платёжной системы), осуществляющий некую вредоносную операцию (например, перевод денег на счёт злоумышленника).

Для осуществления данной атаки жертва должна быть аутентифицирована на том сервере, на который отправляется запрос, и этот запрос не должен требовать какого-либо подтверждения со стороны пользователя, которое не может быть проигнорировано или подделано атакующим скриптом.

# Описание эксперимента

## **Формат данных**

Для статистической оценки количества и типа уязвимостей необходимо выбрать источник информации. В контексте задачи было решено взять 1000+ веб-сайтов из списка «Alexa – TOP 1M», что гарантирует достаточно посещаемые сайты, но и статистическую устойчивость результата (сайты были выбраны из последней четверти этого миллиона, для того, чтобы исключить очевидно сильные в плане защиты сайты – Google, Facebook, etc.).

Все сайты из списка были протестированы с помощью OWASP-test tool, VSU-CVSS, NIKTO-scanner, DIRB. Результаты сведены в таблицу уязвимостей, и, с помощью оценок OWASP и CVSS, посчитаны их бальные оценки. Далее результаты разделены по категориям критичности уязвимостей и их типам.

## **Практическая реализация системы тестирования**

Практическая реализация представляет собой набор скриптов, передающих информацию друг-другу, начиная с ссылки на сайт и заканчивая перечнем, количеством и бальной оценкой обнаруженных уязвимостей.

Все системы тестирования возвращают CVE – коды уязвимостей, которые можно посмотреть в каталоге. В нем же указаны типы этих уязвимостей и условия эксплуатации. С помощью этих данных можно провести оценку CVSS и получить количество баллов критичности уязвимости.

Необходимо учесть топ OWASP, поэтому введен дополнительный коэффициент. Наличие уязвимости в актуальном топе OWASP – увеличивает её критичность, связано это с тем, что обычно разработчики проверяют сайты открытыми утилитами тестирования по данному топу – а значит наличие выбранной уязвимости еще более критично.

# Статистика

Вся статистика, обычно, сводится к подсчету открытых, или собранных каким-либо образом данных. Но что делать, если разработчики, боясь собственной компрометации, утверждают, что их ресурс на 100% безопасен? Очевидно, что это ложь. Еще большей ложью можно назвать такие заявления от людей, которые еще и знают или хотя бы догадываются, что на самом деле это не так. Дело-то все в том, что обычному разработчику сложно признаться в наличии существенных брешей в безопасности до момента, когда они уже будут обнаружены. Ну и кроме того, многие действительно не догадываются о их существовании, а раскошелиться на аудит компания/разработчик просто не в состоянии. Ну и самая популярная проблема, хотя иногда и вынужденная (начальством, сроками, подставьте свое) – концепция «\*\*\*\*, \*\*\*\* - и в продакшн!», которая, конечно, росту безопасности не способствует. На фоне всего этого крайне интересно выглядит инфографика, которая сейчас будет представлена.

Для изучения было выбрано 1000 случайных веб-приложений из списка «Alexa – top 1M», которые были подвергнуты автоматическому тестированию с помощью четырех самых распространенных сканеров.

Типы угроз классифицированы с помощью систем OWASP и TWASC TCv2, а качественная оценка каждой из них – CVSSv2. Всего выявлено больше 10 тысяч уязвимостей разных степеней риска.

Важно учесть, что все системы доступны извне, то есть из Интернета. При этом только 18% этих систем находились в режиме тестирования. Согласно цифрам, предоставленным Positive Technologies – высокой степенью риска обладали только 68% уязвимостей, в 2013 году – 62%, текущие же данные еще хуже: 73% уязвимостей оказались критическими.

Да и вообще говоря, количество уязвимостей на веб-сервис выросло, 32~ уязвимости на веб-приложение, что по сравнению с прошлым годом (29,9) и 2013 (15,6) – еще хуже. А количество критических уязвимостей, которые могут быть обнаружены в автоматическом режиме выросло на ~13%, по сравнению с прошлым годом.

Вывод из всего этого только один, дыр стало больше, а значит находить их стало легче, конечно, не только пентестерам, но и злоумышленникам.

Но самое удивительное заключается в том, что, как и в прошлом году, количество критических проблем в большей степени замечено на «боевых» системах. Еще более страшное – больше всего сайтов с уязвимостями высокой степени риска принадлежит банковской отрасли – таковых в ней 87%. Однако это и логично, во-первых, при написании такого рода приложений большое внимание уделялось финансовой части. А потом техническая теперь страдает. Ну и во-вторых, статистическая погрешность, здесь на графиках указаны данные в процентах от количества уязвимостей, и если все среднего и слабого типа уязвимости обнаружены не были, или были, но в мизерных количествах – то и процент критических уязвимостей резко вырастет.

А вот тот факт, что приложения, написанные на PHP, на самом деле более уязвимые – вовсе не удивителен. Связано это, конечно, с тем, что порог вхождения в язык очень низкий, ну и его популярность в этой связи крайне высока. Более-менее сопротивляться с ним в этом плане может ASP.NET.

С веб-серверами ситуация крайне похожая, NGINX – и самое большое количество ошибок, очень похоже на PHP, не правда-ли? Нет, на самом деле прямые аналогии проводить не совсем честно. Собственно, инструмент-то ни в чем не виноват, и заслуженно занимает первые строчки в топах самых защищенных приложений. Дело все, конечно, в том, кто и как использует приложение.

Ну и кроме того, больше половины потенциальных атак направлены на клиентскую часть – что, конечно, никак не связано с сервером приложений.

Еще более важно учесть тот факт, что количество ошибок, допущенных в системной части приложений, т.е., скорее всего, в исходном коде самими разработчиками, равно более чем 80%, ну а вот остальные уже – некорректной конфигурацией.

А теперь о самих уязвимостях. Самая распространенная, хоть и не выглядящая опасной – fingerprinting, или отпечатки ПО. Она, конечно, сама по себе не очень опасна, но сам факт возможности сбора информации о ПО, которое использует ваша система – должен заставить задуматься достаточно грамотного разработчика. Ведь это, в сущности своей, разведка и есть. А что бывает после разведки? Правильно, активные действие.

А вот наличие XSS в топе говорит об исключительной неграмотности front-end разработчиков, пишущих формы в приложении, и их товарищах на back-end, которые обрабатывают данные из этих самых форм. Ведь всего пару фильтров почти полностью исключают возможность эксплуатации этого типа уязвимостей.

Таблица №1 – топ уязвимостей в зависимости от инструмента.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PHP | Доля, % | ASP.NET | Доля, % | Other | Доля, % |
| XSS | 95 | Fingerprinting | 78 | Fingerprinting | 67 |
| Fingerprinting | 76 | XSS | 44 | C/S Prediction | 67 |
| SQL Injection | 67 | Insuffient Authorization | 44 | XSS | 50 |
| C/S Prediction | 62 | Brute Force | 44 | Brute Force | 33 |
| Abuse of Functionality | 48 | SQL Injection | 33 | Insuffient Authorization | 32 |
| Insuffient Authorization | 43 | C/S Prediction | 33 | SQL Injection | 31 |
| CSRF | 43 | CSRF | 33 | CSRF | 30 |
| URL Redirector Abuse | 43 | Abuse of Functionality | 22 | URL Redirector Abuse | 29 |
| Brute Force | 38 | Information Leakage | 22 | Information Leakage | 29 |
| Information Leakage | 33 | Brute Force | 22 | Brute Force | 26 |

Уязвимость того же уровня риска Credential/Session Prediction. Совсем просто – возможность узнать идентификационный номер любого пользователя, обладая каким-то набором начальных параметров. На практике, злоумышленники могут, обладая информацией о генерации этого самого номера, присвоить себе ваш номер, и на время, получить доступ к системе с вашими правами. Возможно это в следствие различного рода утечек исходного кода, или другой компрометирующей информации, которая позволяет понять алгоритм работы таких систем (генерации идентификационных номеров).

Таблица №2 – соотношение сервер- и администратор-зависимых уязвимостей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| % | Apache | IIS | Nginx | Traefik |
| Fingerprinting | 80 | 78 | 79 | 5 |
| Brute Force | 40 | 44 | 36 | 8 |
| Information Leakage | 30 | 20 | 29 | 40 |
| Server Misconfiguration | 30 | 20 | 21 | 10 |
| Predictable Res. Location | 20 | 8 | 14 | 10 |

Относительно ровные результаты Apache, IIS и Nginx связаны с тем, что и их количество не слишком-то отличается. А вот в последнем случае, аномально низкие результаты проверки по fingerprinting и brute force связаны со стандартной конфигурацией. Она включает в себя защиту от сканирования, собственно traefik абсолютно не выводит информацию о маршрутизируемых системах, и защиту от DoS атак, некий сканер с накоплением, перестающий принимать пакеты от клиента, в случае их аномального количества.

Остальные результаты относительно ожидаемы, утечки информации, ошибки конфигурации, нахождение ресурсов в открытом месте – все эти проблемы связаны с недостатком знаний и опыта у администраторов, занимающихся настройкой указанных приложений. Однако это-то и важно, ведь опасность представляет собой не сам метод атаки, а вообще возможность его эксплуатации в веб-приложении.

# Заключение

Надо сделать вывод, что уровень защищенности крайне низкий. Более того, по сравнению с данных PT, за 2013 год, ситуация стремительно становится хуже – доля сайтов с критическими уязвимостями растет, а эти самые уязвимости находятся полностью автоматическими сканнерами.

Связано это все с «отсутствием последовательного подхода к обеспечению безопасности приложений на всех стадиях жизненного цикла».

Интересный факт, банковско-финансовые системы имеют 87% критических уязвимостей, а системы электронной коммерции 39%, хотя и являются смежными областями.

# Список литературы

1. Common Vulnerability Scoring System SIG [Электронный ресурс] : URL: http://www.gilpravo.ru (дата обращения: 13.06.2017).
2. OWASP [Электронный ресурс] : URL: https://www.owasp.org/index.php/Main\_Page (дата обращения: 13.06.2017).
3. OWASP Top Ten Project [Электронный ресурс] : URL: https://www.owasp.org/index.php/Category:OWASP\_Top\_Ten\_Project (дата обращения: 13.06.2017).
4. Positive Technologies [Электронный ресурс] : URL: https://www.ptsecurity.com/ru-ru/ (дата обращения: 13.06.2017).