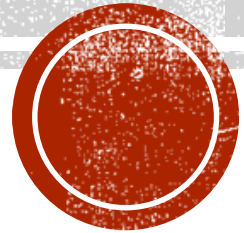
ПОСТРОЕНИЕ МОДЕЛИ ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ ОБНАРУЖЕНИЯ УЯЗВИМОСТЕЙ АВТОРИЗАЦИИ



Анатолий Василенко

621 группа

Научный руководитель:

Гамаюнов Денис Юрьевич

2017

ЦЕЛИ И ЗАДАЧИ

- Цель:
  - Целью работы является разработка алгоритма автоматического построения модели, отражающей функциональные возможности веб-приложения, для дальнейшего обнаружения такого типа уязвимостей как уязвимости авторизации.
- Задачи:
  - ***Исследование*** существующих ***методов*** автоматического построения моделей веб-приложений.
  - ***Разработка метода*** построения модели веб-приложения, отражающей доступную функциональность веб-приложения.
  - ***Описание метода*** обнаружения уязвимостей авторизации на основе разработанной модели.
  - ***Разработка средства***, поддерживающего создание модели для веб-приложения.

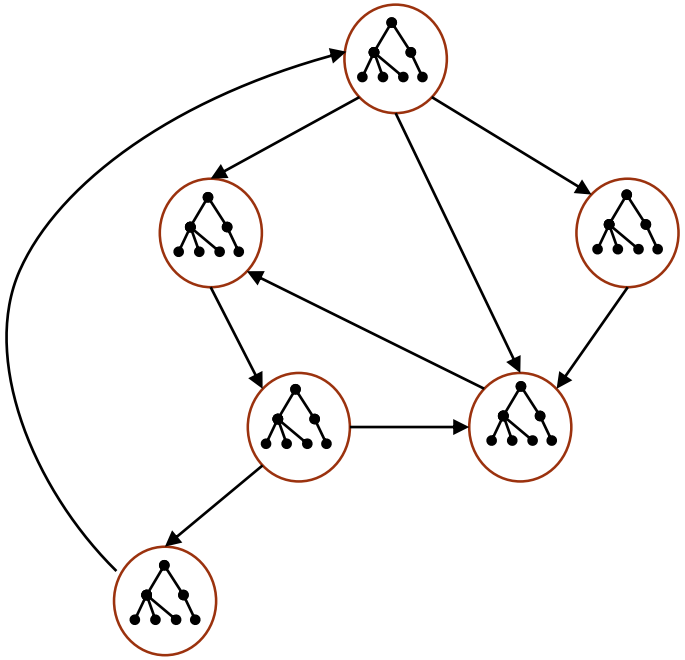
АВТОМАТИЧЕСКОЕ ИССЛЕДОВАНИЕ ВЕБ-ПРИЛОЖЕНИЙ

- Существующие веб-краулеры: *CrawlJax, htcap, ReRIA(CReRIA, CrawRIA), WebMate, …*
  - анализ статических веб-сайтов (URL-based) / интерактивных веб-сайтов (AJAX)
  - обнаружение контента веб-приложения / обнаружение доступной функциональности веб-приложения
- Недостатки существующих веб-краулеров:
  - отсутствие учёта серверного состояния веб-приложения, или возможности его сбрасывать
  - не рассматривается много-пользовательность

- Целевое веб-приложение исследования:
  - контент веб-приложения динамически изменяется в зависимости от действий пользователей
  - поддержка AJAX – Asynchronous JavaScript and XML
  - взаимодействие с сервером посредством протоколов HTTP/HTTPS
- Анализ методом «чёрного ящика» на основе «представления» веб-приложения

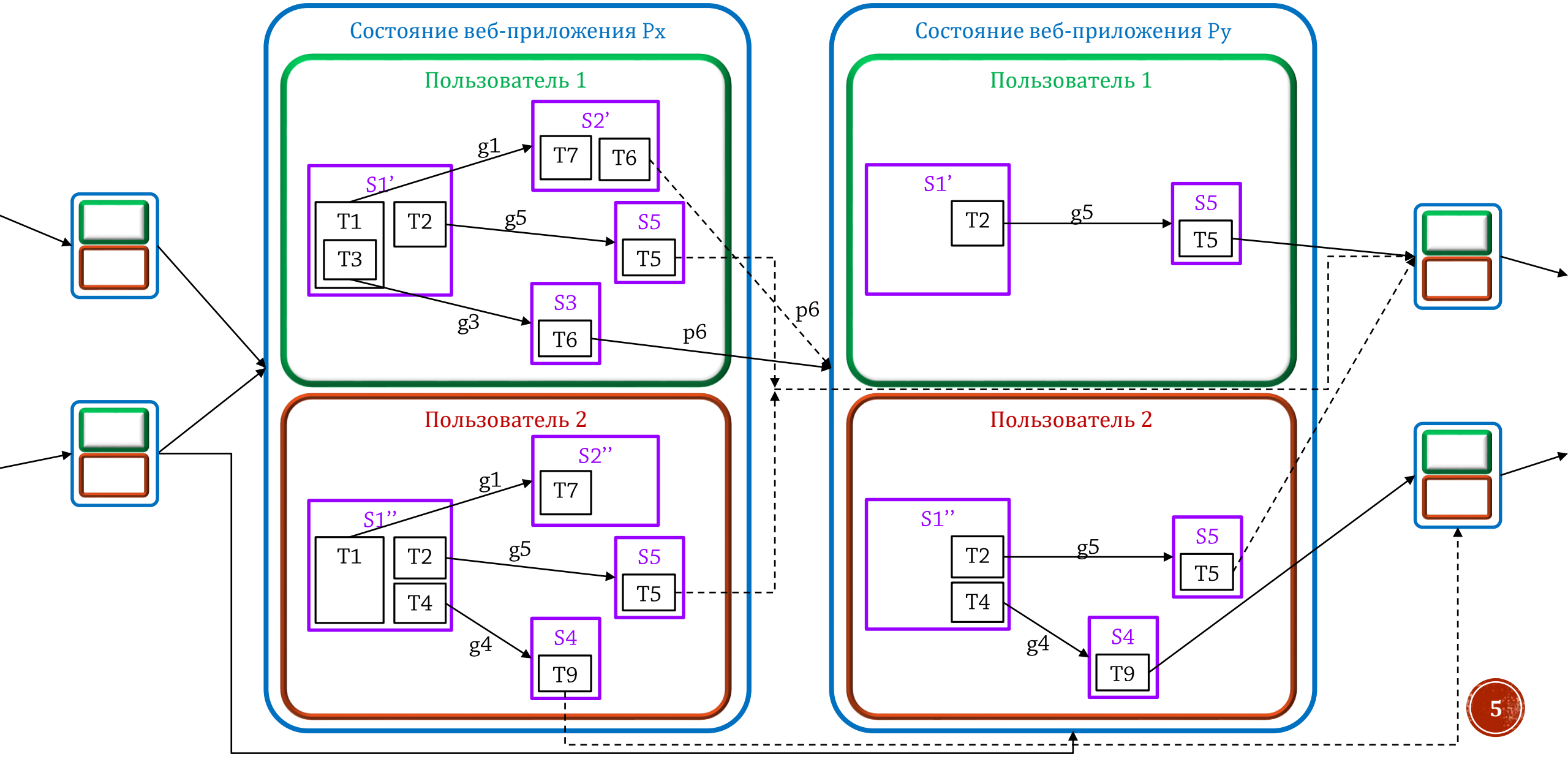
ФУНКЦИОНАЛЬНАЯ МОДЕЛЬ ВЕБ-ПРИЛОЖЕНИЯ

- ***Функциональная модель веб-приложения*** – модель, отражающая все возможные типы действий, доступные некоторым пользователям в различных состояниях веб-приложения.

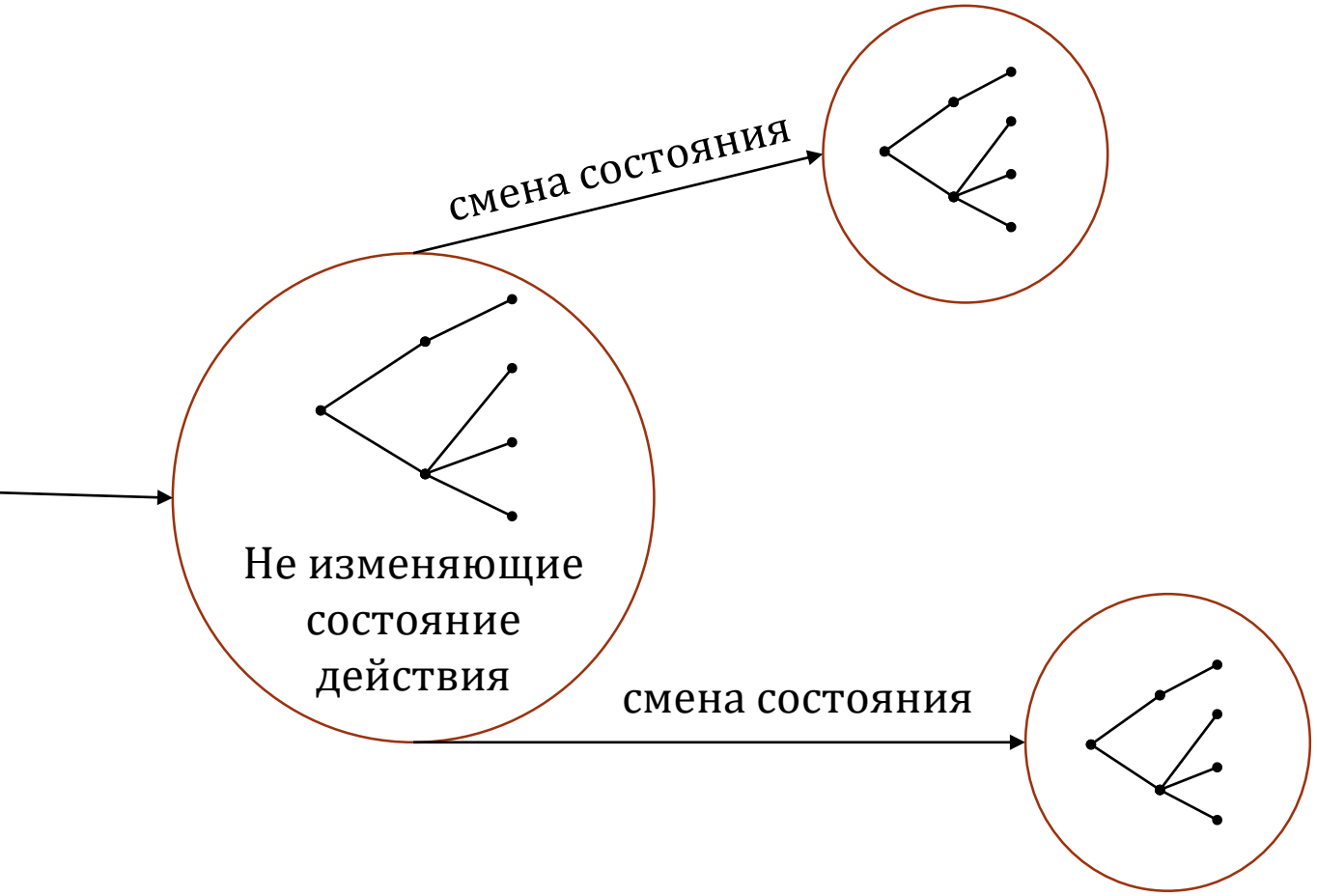


- Моделью является вложенный граф состояний и переходов с выделением шаблонов.
  - ***Шаблон*** – упорядоченное дерево, изоморфное некоторым поддеревьям DOM-представлений веб-страниц, и являющееся «схожими» с ними.
  - Модель веб-приложения:
    - ❖ состояние веб-приложения
      - ❖ пользовательское состояние веб-приложения – модель веб-приложения от имени пользователя, не совершающего действия, изменяющие состояние веб-приложения
        - ❖ состояние веб-страницы
          - ❖ шаблоны веб-страницы

ФУНКЦИОНАЛЬНАЯ МОДЕЛЬ ВЕБ-ПРИЛОЖЕНИЯ



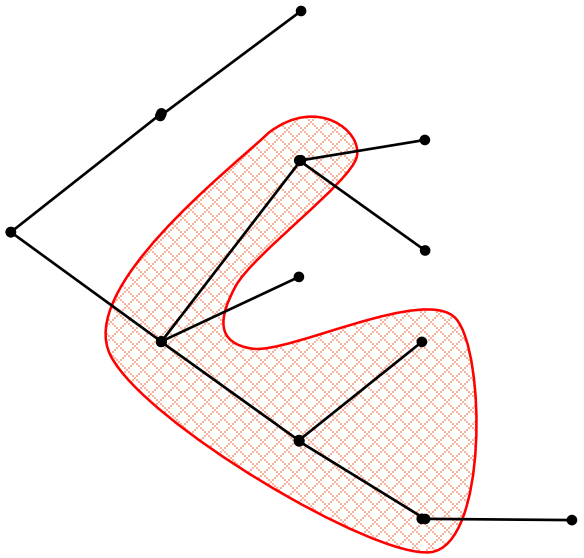
СФЕРА ПРИМЕНЕНИЯ – ТИПЫ ДЕЙСТВИЙ



Предположение о разделении действий пользователя:

- действия пользователя, приводящие к HTTP запросу с методом «GET» будем считать ***не изменяющими*** состояние веб-приложения,
- остальные действия пользователя будем считать ***изменяющими*** состояние веб-приложения.

СФЕРА ПРИМЕНЕНИЯ - ШАБЛОНЫ



- ***Предположение о шаблонах:*** Будем предполагать, что для любого состояния веб-приложения и любой веб-страницы, действиям одного пользователя в рамках одного шаблона соответствует одинаковая функциональность со стороны веб-приложения.

- ***Предположение об отсутствии влияния предыстории:*** Действие пользователя в рамках определённого шаблона при повторном совершении приводит к одному и тому же результату независимо от предварительной последовательности пользовательских действий.

- Перевод веб-приложения в состояние допускающее **интересующее пользовательское действие**:
  - идентификация шаблона, которому принадлежит интересующее действие
  - обнаружение действия, переводящего состояние веб-приложение в состояние, в котором появляется идентифицированный шаблон
  - рекурсивное повторение предыдущих операций до обнаружения последовательности действий, переводящих текущее состояние веб-приложения в необходимое

ИСПОЛЬЗОВАНИЕ ФУНКЦИОНАЛЬНОЙ МОДЕЛИ ВЕБ-ПРИЛОЖЕНИЯ

Обнаружение:

- «точек входа» в веб-приложение.
- уязвимостей типа Stored-XSS.
- уязвимостей авторизации.

УЯЗВИМОСТЬ АВТОРИЗАЦИИ

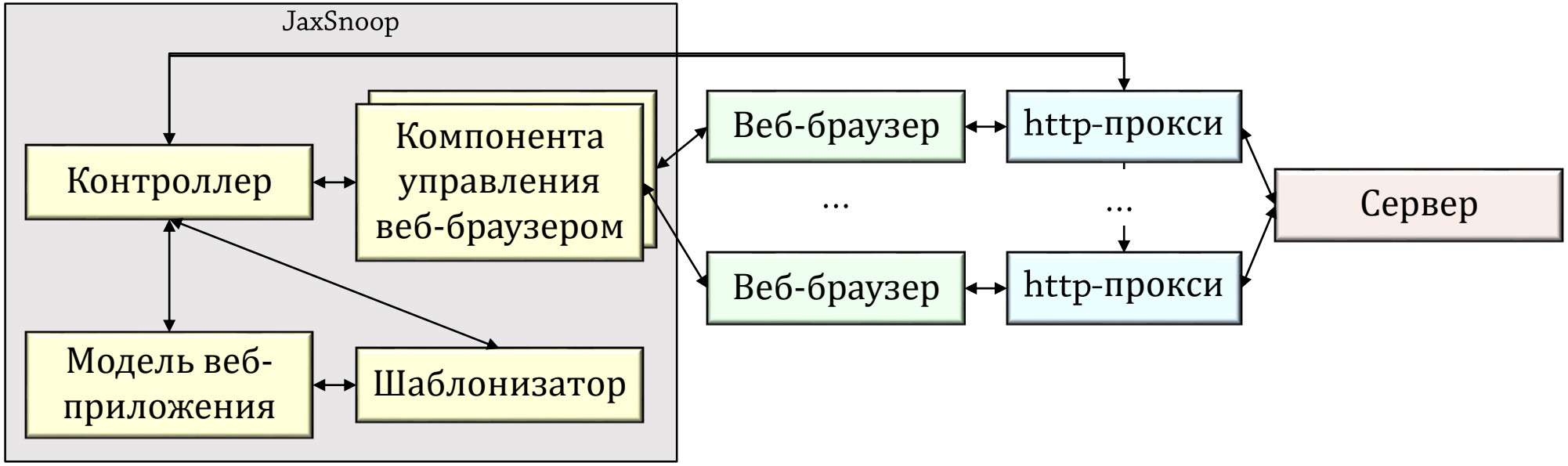
- ***Уязвимость контроля доступа*** *(или* ***уязвимость авторизации****)* – это недостаток в системе, позволяющий конкретному пользователю иметь доступ к некоторому ресурсу или совершать действия, запрещённые политикой контроля доступа.
  - Согласно данным группы OWASP уязвимость авторизации занимает 4-е место по распространённости (https://www.owasp.org/index.php/Top_10_2017-Top_10)

- ***Предположение***: Действия, которые доступны пользователю из его веб-интерфейса являются разрешёнными, остальные действия считаются запрещёнными.
  - «Обнаружение уязвимостей авторизации в веб-приложениях» Носеевич Г.М. 2012 г.
  - Недостаток: необходимо вручную обойти состояния веб-приложения для построения карты сайта.

- ***Детектирование уязвимостей авторизации*** происходит посредством обнаружения всех пользовательских действий, которые для некоторого состояния веб-приложения доступны одному пользователю, но запрещены другому, и совершения этих действий с подменой аутентификационных данных-cookies.

СРЕДСТВО JAXSNOOP

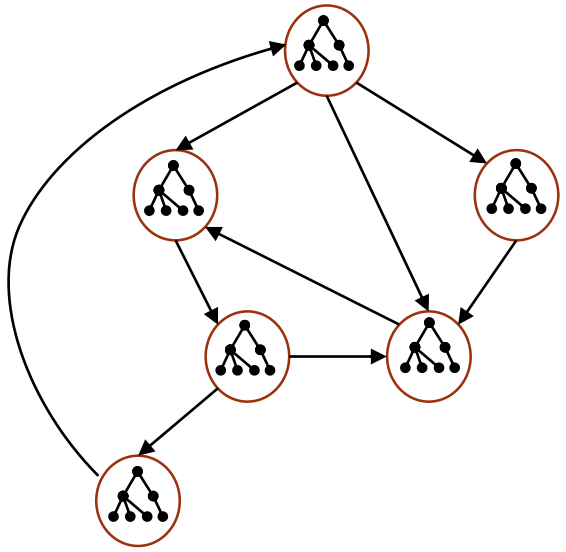
- JaxSnoop:
  - NodeJS – программная платформа, транслирующая JavaScript в машинный код, и предоставляющая API для работы с устройствами ввода-вывода и подключения внешних библиотек.
  - Selenium-webdriver – набор библиотек, позволяющий автоматически управлять различными браузерами.
  - ClientProxy - HTTP/HTTPS веб-прокси, позволяющий перехватывать и изменять запросы и ответы при общении браузера с веб-сервером.



JaxSnoop - http://github.com/phonexicum/jaxsnoop

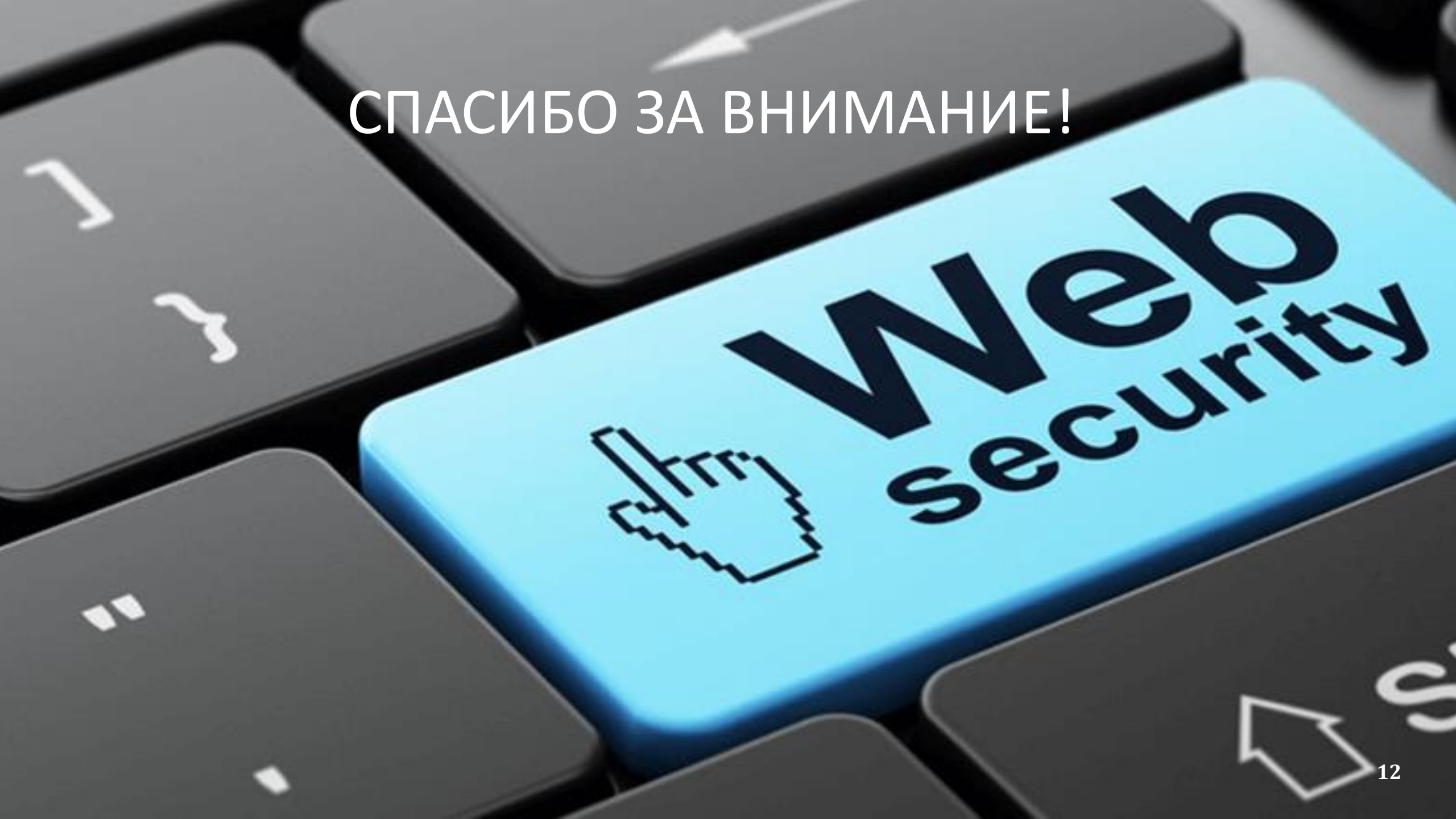
ClientProxy - https://github.com/phonexicum/ClientProxy

ЗАКЛЮЧЕНИЕ



- Предложена модель веб-приложения, учитывающая много-пользовательность и наличие состояния на стороне сервера. Модель позволяет ***увеличить уровень автоматизации при поиске уязвимостей*** в веб-приложениях.
- Реализовано ***средство JaxSnoop***, позволяющее строить модель веб-приложения в соответствии с предложенной концепцией.
- Применимость реализованного средства ***исследована*** для таких веб-приложений как «pyforum» и «easy JSP forum».

СПАСИБО ЗА ВНИМАНИЕ!



ДЕТЕКТИРОВАНИЕ УЯЗВИМОСТЕЙ АВТОРИЗАЦИИ

- Для каждого состояния веб-приложения обнаружим множество действий, доступных одному пользователю и недоступных другому.
- Для каждого обнаруженного действия совершим его подменив аутентификационные данные-cookies пользователя с одного на другого.
- После совершения действия зафиксируем успешность или не успешность действия.
  - HTTP код ответа 400 или 500, или несовпадение классов ответов на настоящий и поддельный запрос.
  - Сравнение состояний веб-приложения (состав и количество шаблонов), после настоящего и поддельного запроса.
  - Рассчёт метрики расстояния (расстояние Левенштейна) HTTP-ответов на настоящий и поддельный запрос пользователя.
  - → В случае успеха зафиксируем уязвимость авторизации.
- Восстановим состояние веб-краулера и веб-приложения.