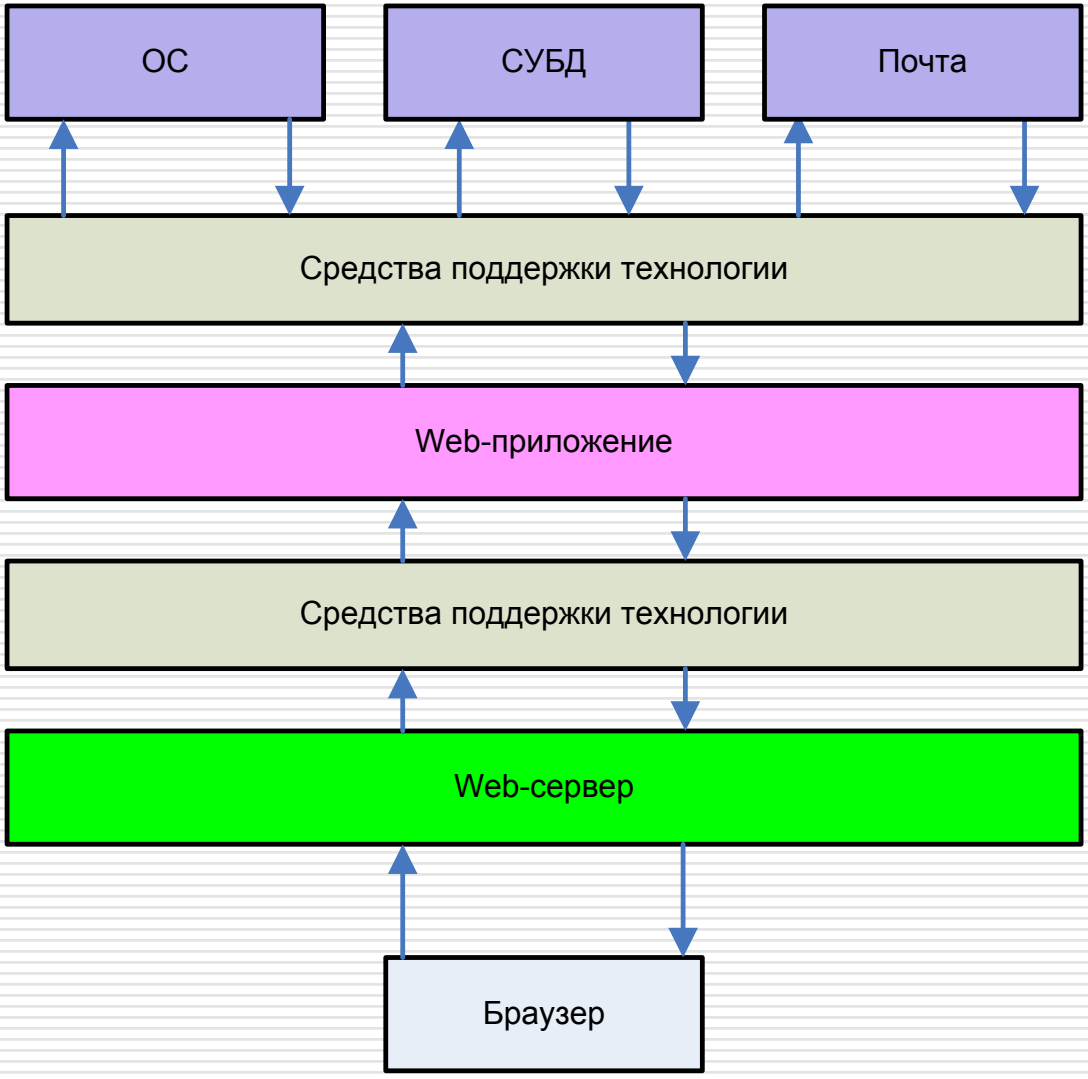
Безопасность приложений

Взаимодействие с внешними сервисами

Схема работы модуля типичного веб-приложения



Взаимодействие с внешними компонентами

- Веб-приложение взаимодействует с внешними подсистемами: СУБД, LDAP, интерпретатор ОС, почтовый демон, браузер, файловая система и т.п.
- У большинства подсистем свой язык: SQL, Shell/CMD, SMTP, HTML и Javascript у браузера, XPath и т.п.
- Обращения веб-приложения во внешние подсистемы параметризуются
  - есть константная часть
  - есть переменная часть, значение которой формируется динамически с участием пользовательских данных
  - обработка запроса http://newsfeed.us.to/news.php?id=17 может содержать оператор вида

    mysql_query(“SELECT * FROM news WHERE id = ”.$id);

Некорректная обработка входных данных

- В каждом языке определены ключевые слова, разделители, правила оформления секций данных:
  - SELECT * FROM news WHERE author = '**petand**'
  - View first
  - grep -R "**search string**" *
- В каждом запросе или команде к внешней подсистеме есть контекст команд и есть контекст данных
- Injection-атаки: внешний пользователь может выйти из контекста данных в контекст команд:
  - SELECT * FROM news WHERE author = '**petand' or sleep(5) --** '
  - <a href="/show?id="**1"><script>alert(1)</script>View first
  - grep -R "**search string" 1; echo "p0wned" #**" *

Примеры Injection-атак

- **SQL-операторы**
  - данные не обрабатываются при построении SQL-запроса
- **HTML-разметку**
  - данные не обрабатываются при построении HTML
- **Javascript-операторы**
  - данные не обрабатываются при построении JSON-объекта
- **CSS-директивы**
  - данные не обрабатываются при построении CSS-правил
- **HTTP-заголовки**
  - данные не обрабатываются при построении ответного заголовка (например, Location)
- **XPath-операторы**
  - данные не обрабатываются при построении XPath-запроса

Взаимодействие с внешними подсистемами - SQLi

SQL injection

- Терминология
  - SQL injection – это название *атаки*
  - недостаток: возможность внедрения операторов SQL
  - причина: некорректная обработка входных данных
- Определение
  - пользователь может изменить структуру SQL-запроса
- Возможные последствия
  - нарушение конфиденциальности/целостности данных
  - нарушение доступности (DROP DATABASE …)
  - нарушение целостности приложения (в т.ч. операций)
  - повышение привилегий на сервере

Пример

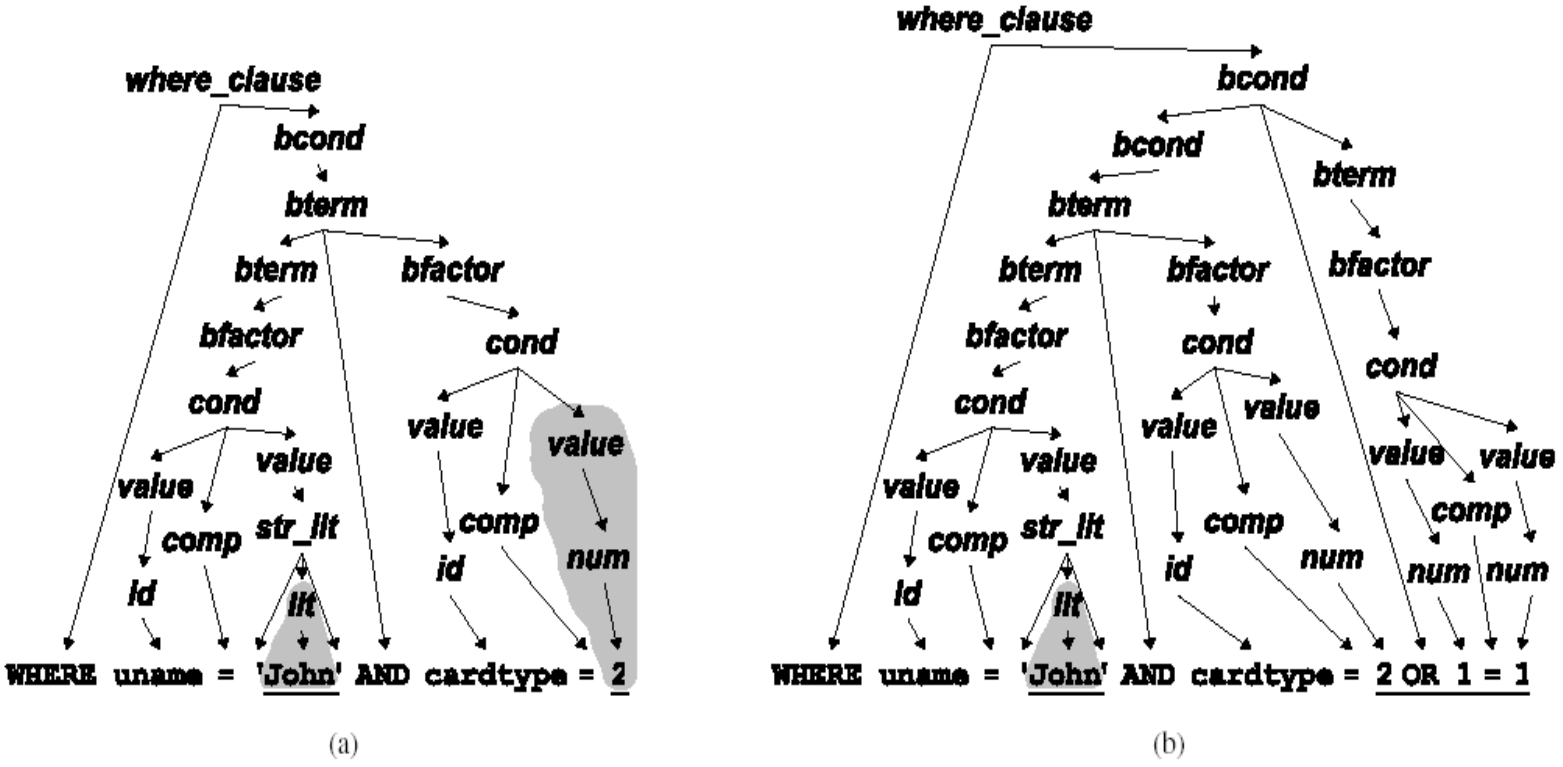
```
$username =  $_POST["username"];
$password =  $_POST["password"];
$query = "SELECT * FROM authuser WHERE
        uname='$username' AND passwd=MD5('$password')";
$result = mysql_query($query);
if ($row = mysql_fetch_array($result, MYSQL_BOTH)) {
        uid = $row["uid"]; //login successful
} else {
        //report error
}
mysql_free_result($result);
```

- First & Second order SQLi

Иллюстрация определения SQLi

SELECT * FROM DB WHERE uname=‘John’ AND cardtype=2

SELECT * FROM DB WHERE uname=‘John’ AND cardtype=2 OR 1



Работа с СУБД: типичный workflow

- Получение данных из HTTP-запроса
- Обработка данных на корректность
- Формирование SQL-запроса
  - DML-запросы, SELECT
- Получение результата SQL-запроса и/или обработка исключительных ситуаций
  - SELECT одной записи, многих записей
- Формирование HTTP-ответа
  - в ответ попадает выборка из СУБД
  - ответ зависит от успешности запроса
  - ответ не зависит от результата SQL-запроса

Методы обнаружения

- Вызвать ошибку (синтаксическую или runtime)
  - ошибка показывается
  - ошибка не показывается
    - надо различать страницы: успешный запрос, неуспешный корректный запрос, некорректный запрос
- Изменить размер выборки
  - and 1=1 vs and 1=0 или or 1=1 vs and 1=1
  - ' and '1'='1 vs ' and '1'='0 или ' or '1'='1 vs ' and '1'='1
- Вызвать задержку
  - MySQL sleep(); MSSQL WAITFOR DELAY; heavy queries
- Вызвать события out-of-band
  - ping, запрос файлов по HTTP, DNS

Demo: обнаруживаем SQLi

Основные контрмеры

- Предотвращение
  - использование prepared statement

```
$stmt = $dbh->prepare("INSERT INTO TAB(name, value) VALUES (?, ?)");
$stmt->bindParam(1, $name);
$stmt->bindParam(2, $value);

$name = 'one';
$value = 1;
$stmt->execute();
```

- Минимизация последствий
  - реализация least privilege для подключений к СУБД
  - вынесение сервера СУБД на отдельный узел
  - контроль целостности приложения

Типичный workflow эксплуатации

- Определить доступные базы
- Узнать имена таблиц в каждой базе
- Узнать количество, имена и тип столбцов
- Узнать содержимое каждой таблицы
- Обычно в СУБД есть системные таблицы с мета-информацией

Demo: мета-информация в MySQL

MySQL (>= 5.0)

- Получение схемы:

  SELECT schema_name FROM information_schema.schemata

  SELECT table_name FROM information_schema.tables
  WHERE table_schema = …

  SELECT column_name FROM information_schema.columns
  WHERE table_schema = … AND table_name …

  SELECT column_name FROM table_name – сливаем контент

- Получение общего числа записей:

  SELECT count(*) FROM … WHERE …

- Получение записей по одной (0 <= j < count(*)):

  SELECT column_name FROM schema_name.table_name
  WHERE … LIMIT j,1

Методы эксплуатации (1 из 2)

- Ошибка СУБД выводится в HTTP-ответ
  - вызывать runtime-ошибки

SELECT name, desc FROM names WHERE uid = **1 UNION (SELECT count(*), concat(*string*,floor(rand(0)*2)) AS x FROM table GROUP BY x)**

  - ***string*** может быть version() или (SELECT schema_name FROM information_schema.schemata LIMIT j,1)
  - самостоятельное исследование: почему rand(0)?
- Выборка выводится в HTTP-ответ
  - использовать UNION

SELECT name, desc FROM names WHERE uid = **1 UNION SELECT '', schema_name FROM information_schema.schemata**

Demo: Proof of Concept

Методы эксплуатации (2 из 2)

- Ни ошибка, ни выборка не выводится в ответ
  - задавать вопросы да/нет
    - ловим задержку или различие в страницах
- Определение целого (метод бисекции):
  - SELECT * FROM users WHERE uid = 1 **and (SELECT count(*) FROM information_schema.schemata) < 1000 --**
- Определение строки:
  - SELECT * FROM users WHERE uid = 1 **and (SELECT length(column) FROM table LIMIT j,1) < 100 --**
  - SELECT * FROM users where uid = **1 and (SELECT substr(column,i,1) FROM table LIMIT j,1) > 'a' --**
- Автоматизация: sqlMap, Toolza

Demo: автоматизация!

Домашнее задание

- Максимально далеко пройти в лабораторной работе от Дмитрия Евтеева
  - http://download.securitylab.ru/PT_SQL_Injection_MIFI-LAB.rar
  - задания ищите на http://course.secsem.ru/tasks
- Установить HacMe Bank v2 и найти там SQLi
  - какие технические цели можно достичь, используя эту уязвимость? Перечислите их и опишите методы
- Ссылки:
  - Книга «SQL Injection Attacks and Defense»
  - Блог Reiners'a: http://websec.wordpress.com/
  - http://h.ackack.net/cheat-sheets/mysql

Корректная обработка пользовательских данных

- Проверки на клиенте должны дублироваться на сервере
- Любые данные, контролируемые клиентом, являются потенциально опасными
- Валидация. Единственно верным подходом к обработке входных данных является подход “accept known good”
  - black-listing’у отказать!!!!
  - в частном случае – приведение к типу (дата, число, boolean)
- Валидация входных данных относительно белого списка не всегда может гарантировать безопасность
  - входные данные после валидации могут содержать служебные символы и разделители языка служебного компонента
  - => необходимо осуществлять экранирование или кодирование
  - например, O’Neal в SQL-операторе должен стать O\’Neal
  - (a + b) < c в HTML должно быть (a + b) < c

Внешняя подсистема: XML

- Temporary redirect to
  - http://www.owasp.org/images/5/5d/XML_Exteral_Entity_Attack.pdf
  - http://www.owasp.org/images/1/18/OWASP_JOU_XML_DTD_Attacks.pptx
  - www.xakep.ru/magazine/xa/160/xa_160.pdf (с 18-ой странице)

Server Side Request Forgery

- Сервер делает запросы по URL, который контролирует пользователь
  - XXE
  - загрузка аватарки по URL
  - перепост с preview
  - общение с backend-серверами
  - обработка файлов со структурой, включающий удаленные ресурсы (HTML2PDF, например, ODF)
- Варианты:
  - можно/нельзя контролировать тело запроса
  - можно/нельзя прочитать ответ
- Т.е. мы управляем HTTP-клиентом на сервере

SSRF: вектора

- Что делать, если вы контролируете не весь URL (например, все, кроме схемы)
  - использовать редиректы (http:// -> file://)
  - tftp – это UDP-протокол
  - gopher позволяет писать практически любой текст в сокет (т.е. сделать POST-запрос)
- Вектора:
  - сканировать сеть за уязвимым узлом
  - делать dirbusting
  - получить доступ к сервисам, слушающим 127.0.0.1
    - memcached / управляющие порты веб-сервера
  - в общем случае – exploitation of trust

Вопросы?