Визитка

МАКСИМ КЛЕЦКИН, инженер компании «Доктор Веб»

Сервис удаленного тестирования

Удаленный рабочий стол и Linux-консоль в окне браузера

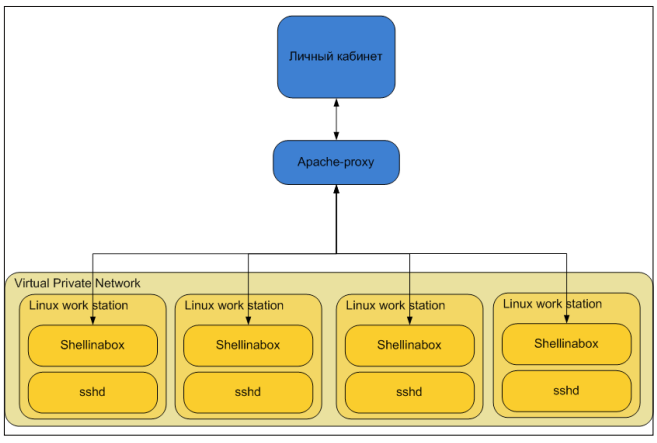
В свое время в компании «Доктор Веб» был создан сервис удаленного тестирования Dr.Web LiveDemo [1]. Сервис позволял тестировать продукты Dr.Web тем клиентам, кто по какой-либо причине не мог развернуть их в рамках своей сети, а также позволял менеджерам и партнерам компании организовывать демонстрации тех же решений

Пользователям сервиса предлагались виртуальные машины нескольких версий Windows (Win2008R2, WinXP, Win7 – в двух последних случаях можно было использовать русские и/или английские версии), а также машины на основе CentOS. В первом случае пользователь получал доступ к рабочему столу по стандартному протоколу RDP, а во втором – к текстовой консоли по SSH.

Сервис работал неплохо, однако клиенты постоянно испытывали трудности с авторизацией. Несмотря на выдаваемую вместе с паролями доступа инструкцию, пытались получить доступ по RDP к консоли Linux. В организациях с повышенными требованиями к безопасности зачастую оказывались закрыты rdp- и ssh-порты. Все это значительно сужало возможности использования в целом востребованного сервиса.

В связи с этим перед нами встала задача сделать Dr.Web LiveDemo более дружелюбным к пользователю. Нужно было добиться того, чтобы от пользователя больше не требовалось знание о существовании каких-либо протоколов доступа и была бы исключена установка любых утилит (включая putty).

Рисунок 1. Схема передачи ssh в Личный кабинет



Самым оптимальным решением этой проблемы виделось объединение всех доступов (RDP и SSH) в один личный кабинет, где пользователь сможет нажатием кнопки попасть на нужную ему машину (будь то удаленный рабочий стол Windows или ssh-консоль Linux). Задача была понятна, но как ее решать, поначалу ясности не было никакой. Поскольку ОС Linux мне более знакома, то мы начали работу с этой системы.

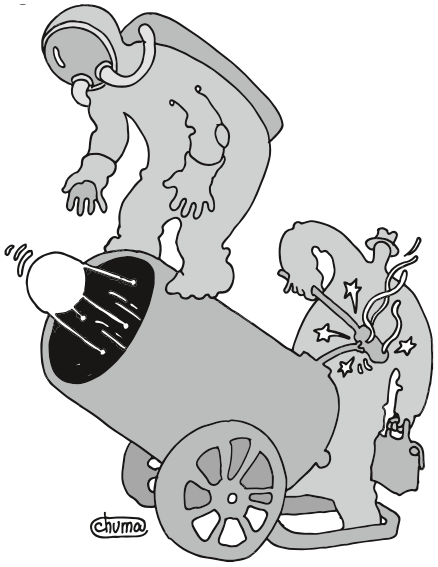
Первой идеей было попробовать создать веб-страницу, которая будет за счет AJAX общаться с сервером и транслировать команды со страницы и ответы системы на них. Это решение было несколько примитивно, но вполне жизнеспособно.

Однако эргономика и удобство подобной реализации были довольно сомнительны. Скорее всего пришлось бы писать полноценный веб-ssh-клиент, и эта работа длилась бы куда больше полугода.

К счастью, в Linux крайне мало мест, к которым человек еще не приложил свою руку, поэтому мы решили узнать, что и как делали до нас. Далеко ходить не пришлось – как выяснилось, веб-ssh уже давно повсеместно используется, и разработок в этом направлении очень много. Оставалось только выбрать наиболее подходящий инструмент для проекта. После тестирования различных реализаций выбор пал на Shell In a Box.

Shell In a Box представляет собой демона, написанного на C++, который поднимает на выбранном порту веб-страничку. При этом он быстро работает и не перегружает страницу javascript, что оказалось неожиданным, но весьма приятным бонусом.

Не обошлось и без ложки дегтя: оказалось, что Shell In a Box не может соединяться с удаленными системами от имени root-пользователя, для запуска Shell In a Box с привилегиями суперпользователя он должен запускаться локально. Для выхода из этой неприятной ситуации нужно было всего лишь организовать единый proxy (на базе nginx или apache), а требуемые системы можно различать по url. В результате ssh передается в Личный кабинет по следующей схеме (см. рис. 1).



Сервис позволяет клиентам как тестировать продукты компании, так и организовывать демонстрации решений

С Linux все оказалось очень просто, но было абсолютно непонятно, как сделать что-то подобное с удаленным рабочим столом Windows. Несколько лет назад это было бы вообще невозможно (пришлось бы, наверное, что-либо организовать на основе Adobe Flash), но мы уже живем при HTML5 и CSS3, так что был выбран проект под названием Guacamole [2].

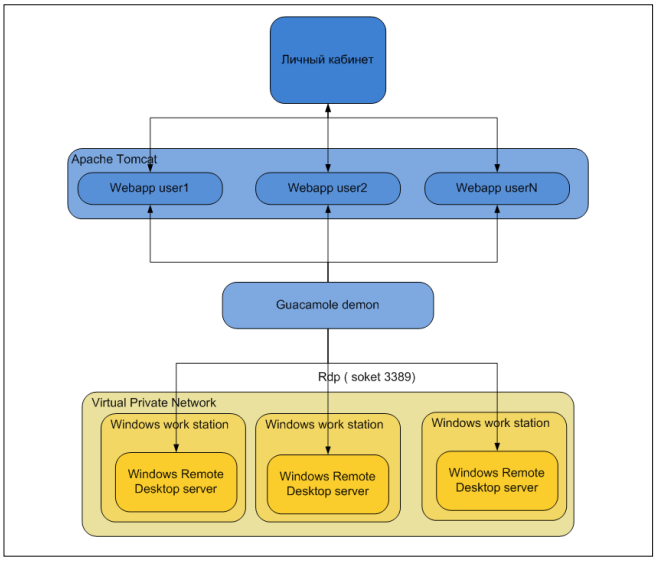
Проект был довольно молодой, но находился уже на той стадии, когда он мог помочь в решении нашей проблемы. Вообще Guacamole – это двухсоставное приложение. В качестве frontend выступает java webapp, а в качестве backend – написанный на C++ демон, который соединяется с хостами по vnc или freeRDP.

После осмотра и тестирования выяснилось, что демон очень гибкий по настройкам соединений, при этом все параметры записываются в конфигурационный файл, имеющий XML-формат, что позволит нам изменять параметры соединения путем замены нужных строк (поскольку в сервисе каждому новому пользователю выдается новый пароль доступа к виртуальным машинам, то соответственно и Guacamole должен их знать).

Вот пример блока авторизации для user2:

```
<authorize username="user2" password="Ke6jaicha" >

    <connection name="Windows 7 RUS">
        <protocol>rdp</protocol>
        <param name="hostname">192.168.203.2</param>
        <param name="port">21122</param>
        <param name="password">Ke6jaicha</param>
        <param name="username">drweb\user</param>
    </connection>

    <connection name="Windows XP SP3 RUS">
        <protocol>rdp</protocol>
        <param name="hostname">192.168.203.2</param>
        <param name="port">21120</param>
        <param name="password">Ke6jaicha</param>
        <param name="username">drweb\user</param>
    </connection>

    <connection name="Windows server 2008 SP1 R2 x64 eng">
        <protocol>rdp</protocol>
        <param name="hostname">192.168.203.2</param>
        <param name="port">21150</param>
        <param name="password">Ke6jaicha</param>
        <param name="username">drweb\user</param>
    </connection>

    <connection name="Windows 7 ENG">
        <protocol>rdp</protocol>
        <param name="hostname">192.168.203.2</param>
        <param name="port">21125</param>
        <param name="password">Ke6jaicha</param>
        <param name="username">drweb\user</param>
    </connection>

    <connection name="Windows XP SP3 eng">
        <protocol>rdp</protocol>
        <param name="hostname">192.168.203.2</param>
        <param name="port">21123</param>

        <param name="password">Ke6jaicha</param>
        <param name="username">drweb\user</param>
    </connection>

</authorize>
```

Рисунок 2. Схема соединения с Windows-машинами



Для изоляции пользователей решено было сделать несколько webapp. Таким образом, вопреки ожиданиям схема реализации трансляции удаленного рабочего стола Windows в Личный кабинет Dr.Web LiveDemo выглядела даже проще, чем для Linux (забегая вперед, отмечу, что так нам казалось на момент выбора решения), вследствие чего была выбрана следующая схема соединения с Windows-машинами (см. рис. 2).

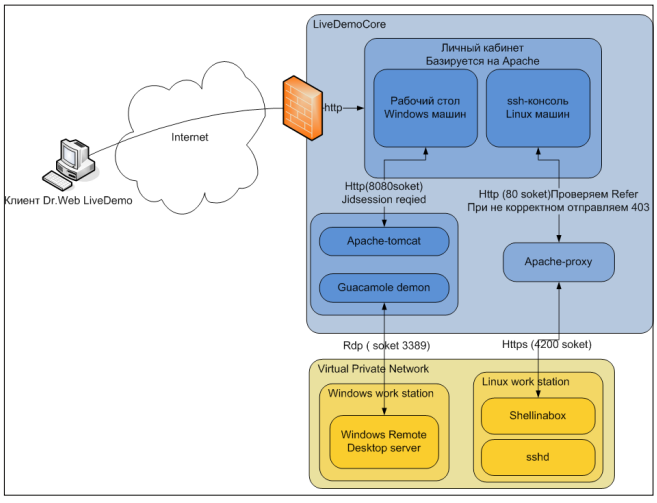
Осталось организовать кроссавторизацию в Личный кабинет (при заходе и авторизации в Личном кабинете пользователь автоматически получает валидный cookie от Tomcat и минует авторизацию в самом Guacamole). Тут вступает в действие особенность самих cookie.

Поскольку Guacamole и Личный кабинет находятся на разных веб-серверах, то мы не можем записать cookie для Guacamole средствами Личного кабинета (если, конечно, не ломать браузер клиента и не использовать уязвимости ОС клиента). Таким образом, у нас было два варианта: переводить Личный кабинет на java или использовать proxy, который будет транслировать данные от tomcat в веб-сервис, поддерживающий Личный кабинет. Мы выбрали второй путь.

Получение cookie было реализовано следующим несложным php-кодом:

```
function login($url,$login,$pass){
   $ch = curl_init();
   // если соединяемся с https
   if(strtolower((substr($url,0,5))=='https')) {
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
   }
   curl_setopt($ch, CURLOPT_URL, $url);
   curl_setopt($ch, CURLOPT_REFERER, $url);
   curl_setopt($ch, CURLOPT_POST, 1);
   curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
   curl_setopt($ch, CURLOPT_POSTFIELDS, ↵
        "username=".$login."&password=".$pass);
   curl_setopt($ch, CURLOPT_USERAGENT, "Mozilla/4.0 ↵
        (Windows; U; Windows NT 5.0; En; rv:1.8.0.2) ↵
        Gecko/20070306 Firefox/1.0.0.4");
   curl_setopt($ch, CURLOPT_HEADER, 1);
   curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
   $result=curl_exec($ch);
   curl_close($ch);
```

Рисунок 3. Реализация Личного кабинета




```
    $result=explode ("\n",$result);
  return $result['4'];
}
```

Но, как всегда, все хорошо бывает только в планах. В ходе тестирования (и традиционно за неделю до назначенного срока релиза) случайно выяснилось, что ни frontend, ни backend не поддерживают русскую раскладку клавиатуры. Пришлось забираться внутрь обоих частей Guacamole и исправлять ситуацию. Для frontend исправить ситуацию было довольно просто: нужно добавить чаркоды русской раскладки в список передаваемых – для этого достаточно изменить одну строчку javascript:

```
    if (codepoint >= 0x0100 && codepoint <= 0x10FFFF)
        //return 0x01000000 | codepoint;
        return codepoint;
```

Что касается демона на C++, пришлось прописывать все соответствия чаркодов согласно таблице. Ниже представлен фрагмент кода дописанного демона:

```
    /* j */
    { .keysym = 0x006a, .scancode = 0x24,
        .clear_keysyms = GUAC_KEYSYMS_ALL_SHIFT },
    { .keysym = 0x043E, .scancode = 0x24,
        .clear_keysyms = GUAC_KEYSYMS_ALL_SHIFT },

    /* k */
    { .keysym = 0x006b, .scancode = 0x25,
        .clear_keysyms = GUAC_KEYSYMS_ALL_SHIFT },
    { .keysym = 0x043B, .scancode = 0x25,
        .clear_keysyms = GUAC_KEYSYMS_ALL_SHIFT },
```

И таких кнопок более 50.

Сборка демона и новая проверка функционала показали, что решение помогло: русские буквы начали восприниматься и передаваться из веб-браузера в Windows-машину и наоборот.

Конечная реализация Личного кабинета получилась такой (см. рис. 3).

Поскольку по условиям работы сервиса веб-сервер вообще не будет испытывать нагрузку (в настоящее время Dr.Web LiveDemo поддерживает до девяти одновременных тестирований, так что это даже нагрузкой сложно назвать), то в качестве основного веб-сервера был выбран хорошо нам известный Apache.

Конфиг Apache:

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    AddDefaultCharset UTF-8
    DocumentRoot /usr/local/www/apache22/data/

    RewriteLogLevel 1
    RewriteLog "/var/log/apache2/rewrite.log"

    <Directory />
        Options FollowSymLinks
        AllowOverride None
    </Directory>

    <Directory /usr/local/www/apache22/data/>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride None
        Order allow,deny
        allow from all
        DirectoryIndex index.php index.html
    </Directory>
```

```
    ProxyRequests off
    Proxyvia on
# Pass for Guac
    Proxypass /user6/rdp http://127.0.0.1:8080/user6
    ProxyPassReverse /user6/rdp http://127.0.0.1:8080/user6
    Proxypass /user2/rdp http://127.0.0.1:8080/user2
    ProxyPassReverse /user2/rdp http://127.0.0.1:8080/user2
    Proxypass /user3/rdp http://127.0.0.1:8080/user3
    ProxyPassReverse /user3/rdp http://127.0.0.1:8080/user3
    Proxypass /user4/rdp http://127.0.0.1:8080/user4
    ProxyPassReverse /user4/rdp http://127.0.0.1:8080/user4
    Proxypass /user5/rdp http://127.0.0.1:8080/user5
    ProxyPassReverse /user5/rdp http://127.0.0.1:8080/user5
    Proxypass /user7/rdp http://127.0.0.1:8080/user7
    ProxyPassReverse /user7/rdp http://127.0.0.1:8080/user7
    Proxypass /user8/rdp http://127.0.0.1:8080/user8
    ProxyPassReverse /user8/rdp http://127.0.0.1:8080/user8
    Proxypass /user9/rdp http://127.0.0.1:8080/user9
    ProxyPassReverse /user9/rdp http://127.0.0.1:8080/user9

    <Proxy http://192.168.3.115:4200/>
        Order deny,allow
        Allow from all
    </Proxy>

    #SSH proxy for user2
    Proxypass "/user2/Linux CentOS 6 Internet Gateway/ssh" ↵
        http://192.168.203.2:4200/
    ProxyPassReverse "/user2/Linux CentOS 6 Internet ↵
        Gateway/ssh" http://192.168.203.2:4200/
    Proxypass  "/user2/Office Shield/ssh" ↵
        http://192.168.203.2:21193/
    ProxyPassReverse "/user2/Office Shield/ssh" ↵
        http://192.168.2032:21193/
    Proxypass  "/user2/Linux CentOS 6 server/ssh" ↵
        http://192.168.203.2:21192/
    ProxyPassReverse "/user2/Linux CentOS 6 server/ssh" ↵
        http://192.168.203.2:21192/
<...>
    ProxyPreserveHost On
ErrorLog ${APACHE_LOG_DIR}/error.log

# Possible values include: debug, info, notice, warn,
# error, crit, alert, emerg.
LogLevel warn
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Минусов у выбранного решения оказалось два.

Во-первых, не все браузеры, используемые потенциальными пользователями сервиса, могут поддерживать вышеописанные технологии – в связи с этим возможность входа на тестовые машины по RDP и SSH была сохранена.

Во-вторых, существенно выросли задержки при наличии антивируса на машине пользователя, так как большинство антивирусов на данный момент проверяют трафик по всем портам. Поскольку при доступе непосредственно из браузера весь трафик, проходящий по протоколу HTTP, будет проверяться применяемой пользователем антивирусной программой, то в случаях замедления вывода изображения пользователю рекомендуется либо отключить на время тестирования проверку HTTP-трафика, либо, если отключение антивируса пользователю запрещено политиками компании, использовать утилиты доступа, переключившись на соответствующий вид страницы.

Ну и в заключение – как это выглядит внешне. Получить доступ к выделенным для тестирования виртуальным машинам можно с любой операционной системы, в том числе с Windows и Linux. Вот так выглядит Личный кабинет в случае, когда доступ возможен как с помощью внешних утилит и сервисов, так и непосредственно из браузера (см. рис. 4).

При выполнении системных требований для подключения достаточно нажать кнопку «Подключиться» напротив описания рабочей станции или сервера.

Если системные требования не выполняются – выводится специальное предупреждение (см. рис. 5).

Вид страницы в таком случае немного меняется (см. рис. 6). EOF

1. Фатеев А. Сервис тестирования продуктов Dr.Web LiveDemo на основе VMware ESXi. //«Системный администратор», спецвыпуск-приложение «Виртуализация», №1, 2010 г. – С. 38 40 (http://samag.ru/archive/article/1057).
2. Проект Guacamole – http://guac-dev.org.

Рисунок 4. Вид Личного кабинета

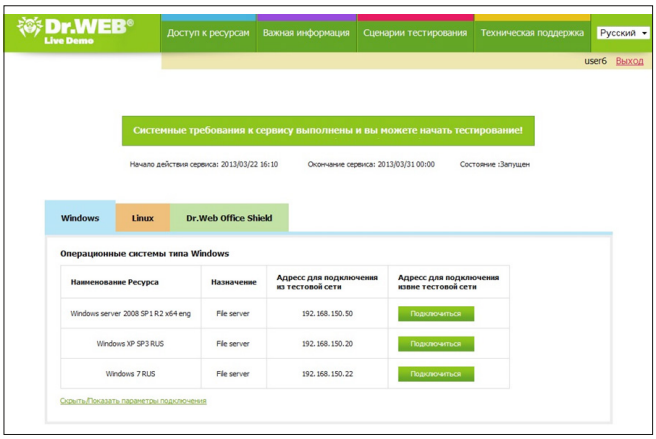


Рисунок 5. Предупреждение о несоответствии системных требований

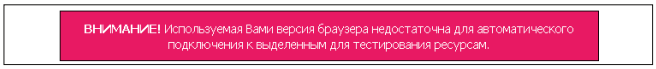


Рисунок. 6. Вид Личного кабинета при выдаче предупреждения