DNS 重绑定攻击实验

1 实验介绍

本实验目的有两个：（1）演示 DNS 重绑定攻击的工作原理；（2）帮助学生获得使用 DNS 重绑定技术攻击的实践经验。在实验配置中，我们有一个仿真的 IoT 设备，可以通过 Web 页面对其进行控制 (许多 IoT 设备也是如此工作的)。大部分 IoT 设备没有强大的保护机制，一旦攻击者可以直接与之交互，就能轻易控制这些设备。

在本实验中的仿真 IoT 设备是一个恒温器，用于控制室内的温度。用户需要能与其进行交互才可以成功设置温度。但由于 IoT 设备部署在防火墙之后，外部计算机无法与 IoT 设备交互，从而无法控制恒温器。为了突破防火墙的保护，攻击代码必须先进入内网，这并不困难。每次用户从内网访问攻击者的 Web 站点时，攻击者的 JavaScript 代码就会在用户的浏览器中执行，因此这个代码实际上是运行在内网中的。但由于浏览器有一个沙盒保护机制，即使攻击者的代码位于内网中，也无法与 IoT 设备进行交互。

本实验的目的是使用 DNS 重绑定攻击来绕过浏览器沙盒保护，进而使得攻击者的 JavaScript 代码可以成功地和 IoT 设备进行交互，从而将恒温器的温度设置为一个非常高的值，超出正常温度范围内。实验包括以下内容：

- DNS 服务器配置
- DNS 重绑定攻击
- 对 IoT 设备进行攻击
- 浏览器的同源策略

相关阅读材料与视频： 关于 DNS 协议与攻击的详细内容可以参考以下材料:

- SEED 教科书, *Computer & Internet Security: A Hands-on Approach*, 3rd Edition, by Wenliang Du. 详情请见 `https://www.handsonsecurity.net`.
- SEED 视频（Section 7）, *Internet Security: A Hands-on Approach*, by Wenliang Du. 详情请见 `https://www.handsonsecurity.net/video.html`.

实验实验环境。 本实验在 SEED Ubuntu 20.04 VM 中测试可行。您可以从 SEED 网站上下载我们预先构建好的镜像并在您自己的电脑上运行 SEED VM。然而，大多数 SEED 实验可以在云端进行，您可以按照我们的说明在云端创建 SEED VM。

2 背景介绍: IoT（物联网设备）

我们的攻击目标是防火墙后面的一个物联网设备，但我们无法从外部直接访问它。我们的目标是让内部用户先运行我们的 JavaScript 代码，然后我们的代码通过 DNS 重绑定攻击来成功地和物联网设备进行交互。

许多物联网设备都有一个简单的内置 Web 服务器，这样用户就可以通过 Web APIs 与这些设备交互。在通常情况下这些物联网设备受防火墙保护，使得它们不能从外部被直接访问。由于这层保护，许多物联网设备并没有部署强大的身份验证机制。如果攻击者能够找到与它们交互的方法，破坏其安全性就是一件很容易的事情。

我们使用一个简单的 Web 服务器来模拟这种易受攻击的物联网设备，该服务器提供两个 API: `password` 和 `temperature`，为了设置室温，我们需要向服务器的 `temperature` API 发送一个 HTTP 请求，该请求需要包含两个参数: 目标温度值和密码。密码是定期更改的，但可以用 `password` API 获取。因此，要想成功设置温度，用户首先需要获得密码，然后在 `temperature` API 中使用该密码。

这里的密码并不是用于身份验证，它只是用于抵抗跨站请求伪造 (CSRF) 攻击。如果没有这种保护，我们使用一个简单的 CSRF 攻击就足够了，没有必要使用复杂的 DNS 重绑定攻击。为了简单起见，我们用的是一个固定的密码，而在现实系统中，密码会定期重新生成的。我们假设攻击者是不知道这个密码的，他们必须通过 `password` API 来获取该密码。

3 使用容器搭建实验环境

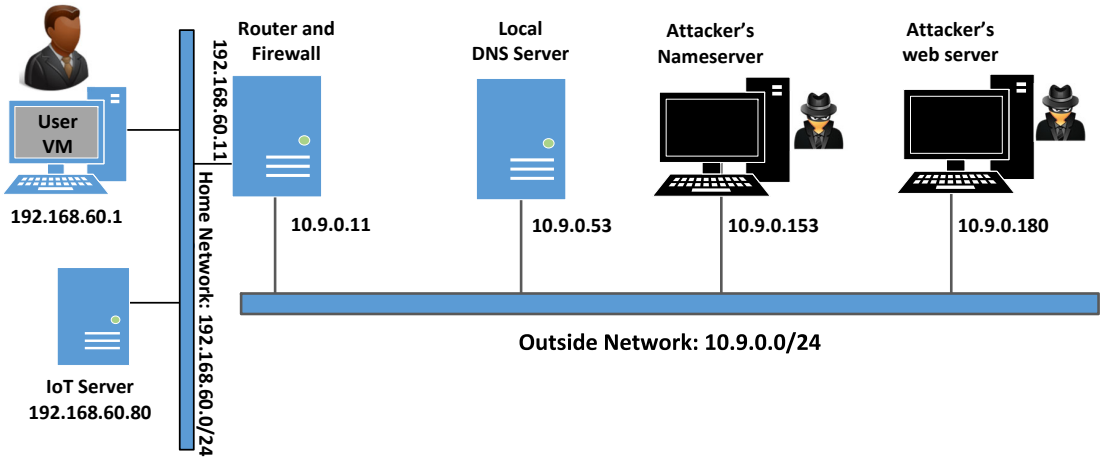

图 1: 实验环境搭建

在本实验中，我们将使用六台机器。实验环境设置如图1 所示。用户机会使用虚拟机，其他均使用容器。在该设置中，我们有两个网络：家庭网络和外部网络。家庭网络模拟典型的家用网络，用户机器和 IoT 设备连接到这个网络，该网络受到路由器上的防火墙的保护。防火墙会拦截所有发往 `192.168.60.80` 的流量。这样，外部机器就无法访问物联网设备。我们还在路由器上设置了一台网络地址转换（NAT）服务器，以便家庭网络中的机器能够访问外部网络。第二个网络模拟外部网络环境。除了路由器之外，

还有三个容器连接到这个网络，其中一个作为本地 DNS 服务器，另外两个分别作为攻击者的域名服务器和 Web 服务器。攻击者拥有 attacker32.com 域名，该域名由攻击者的域名服务器容器托管。Web 服务器托管着一个用于攻击的恶意网站。

3.1 容器设置与命令

请从实验的网站下载 Labsetup.zip 文件到你的 VM 中，解压它，进入 Labsetup 文件夹，然后用 docker-compose.yml 文件安装实验环境。对这个文件及其包含的所有 Dockerfile 文件中的内容的详细解释都可以在链接到本实验网站的用户手册 [1] 中找到。如果这是您第一次使用容器设置 SEED 实验环境，那么阅读用户手册非常重要。

在下面，我们列出了一些与 Docker 和 Compose 相关的常用命令。由于我们将非常频繁地使用这些命令，因此我们在 .bashrc 文件（在我们提供的 SEED Ubuntu 20.04 虚拟机中）中为它们创建了别名。

```
$ docker-compose build  # 建立容器镜像
$ docker-compose up     # 启动容器
$ docker-compose down   # 关闭容器

// 上述 Compose 命令的别名
$ dcbuild       # docker-compose build 的别名
$ dcup          # docker-compose up 的别名
$ dcdown        # docker-compose down 的别名
```

所有容器都在后台运行。要在容器上运行命令，我们通常需要获得容器里的 Shell 。首先需要使用 docker ps 命令找出容器的 ID ，然后使用 docker exec 在该容器上启动 Shell 。我们已经在 .bashrc 文件中为这两个命令创建了别名。

```
$ dockps        // docker ps --format "{{.ID}}  {{.Names}}" 的别名
$ docksh <id>   // docker exec -it <id> /bin/bash 的别名

// 下面的例子展示了如何在主机 C 内部得到 Shell
$ dockps
b1004832e275  hostA-10.9.0.5
0af4ea7a3e2e  hostB-10.9.0.6
9652715c8e0a  hostC-10.9.0.7

$ docksh 96
root@9652715c8e0a:/#

// 注: 如果一条 docker 命令需要容器 ID, 你不需要
//     输入整个 ID 字符串。只要它们在所有容器当中
//     是独一无二的，那只输入前几个字符就足够了。
```

[1] 如果你在部署容器的过程中发现从官方源下载容器镜像非常慢，可以参考手册中的说明使用当地的镜像服务器

如果你在设置实验环境时遇到问题，可以尝试从手册的 “Miscellaneous Problems” 部分中寻找解决方案。

3.2 配置用户虚拟机

我们需要在用户虚拟机上做进一步的配置。

步骤 0. 禁用 Firefox 浏览器的 "DNS over HTTPS" 功能 版本较新的 Firefox 浏览器会默认启用 "DNS over HTTPS" 功能。在该模式下，域名系统（DNS）解析可能不会经过本地 DNS 服务器或/etc/hosts 文件。这会给本实验带来问题，因此我们需要将其禁用。进入 Setting，点击"Privacy & Security" 选项卡；找到 "DNS over HTTPS" 选项，然后选择 off。

步骤 1. 减少 Firefox 的 DNS 缓存时间： 为了并加快 DNS 响应时间，Firefox 浏览器会缓存 DNS 结果。在默认情况下，这个缓存的过期时间为 60 秒。这也意味着我们的 DNS 重绑定攻击需要等待至少 60 秒。为了让实验更快些，我们把时间减少至 10 秒或更少。在用户主机的 Firefox 浏览器的 URL 字段中输入 about:config。通过一个警告页面后，我们将看到一些 preferenc 名称及其值。搜索 dnsCache，找到以下条目并修改它的值:

```
network.dnsCacheExpiration:  将值设置为10（默认情况下是60）
```

完成修改后，我们应退出 Firedox 浏览器然后重启，否则修改不会生效。

步骤 2. 修改/etc/hosts. 我们需要在/etc/hosts 文件中添加以下条目。我们将把 www.seedIoT32.com 作为物联网服务器的名称，其 IP 地址为 192.168.60.80 。需要使用超级用户权限来修改此文件（使用 sudo）:

```
192.168.60.80  www.seedIoT32.com
```

在处理这个文件时，检查是否有任何条目包含 attacker32.com，如果有，请将其删除，因为这些条目可能是我们在做其他 SEED 实验时添加的，它们的存在会对本次实验造成影响。

我们现在可以测试物联网服务器了。在用户虚拟机上的浏览器访问以下网址。如果设置正确，我们应该能看到一个恒温器。我们还可以通过拖动滑块来更改温度设置。请在实验报告中提供一张相应截图。

```
http://www.seedIoT32.com
```

步骤 3. 本地 DNS 服务器 我们需要让用户虚拟机使用环境设置里的本地 DNS 服务器。这可以通过将本地 DNS 服务器设置为解析器配置文件 (/etc/resolv.conf) 中的第一个 nameserver 条目来实现。问题是，虚拟机会使用动态主机配置协议（DHCP）来获取网络配置参数，如 IP 地址、本地 DNS 服务器等。DHCP 客户端会用 DHCP 服务器提供的信息覆盖 /etc/resolv.conf 文件。

一种解决方法是在/etc/resolvconf/resolv.conf.d/head 文件中添加以下条目（在我们的设置中，10.9.0.53 是本地域名系统（DNS）服务器的 IP 地址）:

```
nameserver 10.9.0.53
```

这个 head 文件的内容将被添加到动态生成的解析器配置文件的开头。这个文件本来只有一行注释 (the comment in/etc/resolv.conf comes from this head file)。进行更改后，我们需要运行以下命令以使更改生效：

```
$ sudo resolvconf -u
```

3.3 测试实验环境

配置好用户虚拟机后，使用 dig 命令获得 www.attacker32.com 和 ns.attacker32.com 的 IP 地址。你应该会分别得到 10.9.0.180 和 10.9.0.153。如果你获得的值和这两个有出入，那么说明你的环境没有配置正确。

我们现在可以测试攻击者的网站了。在用户虚拟机上的浏览器访问以下网址，你应该就能看到攻击者的网站。请在实验报告中附上相应的截图。

```
http://www.attacker32.com
```

注意。 在其他 SEED 实验中，我们可能已使用过相同的主机名 www.attacker32.com，因此该名称很可能已映射到不同的 IP 地址。所以，如果你没有看到预期的攻击者网站，就应该检查/etc/hosts 文件，并删除所有包含 attacker32.com 的条目。

4 针对 IoT 设备进行攻击

我们已经做好了攻击 IoT 设备的准备，为了帮助学生更好地理解攻击的原理，我们将攻击拆分为以下几个步骤。

4.1 任务 1. 理解同源策略防护

在此任务中，我们将做一些实验来理解浏览器实现的同源策略保护。在用户主机上，我们访问以下三个 URL。最好是在三个不同的 Firefox 窗口打开这三个页面 (而不是同一个窗口的三个 tab)，这样我们可以同时看到它们。

```
URL 1:  http://www.seedIoT32.com
URL 2:  http://www.seedIoT32.com/change
URL 3:  http://www.attacker32.com/change
```

第一个页面我们能看到当前恒温器设定的温度 (见图 2.a)。它每秒都会从 IoT 服务器获取当前的温度值。我们需要让这个页面保持打开的状态，这样我们可以时刻观察到恒温器当前的温度设定。第二和第三个页面看上去完全相同 (见图 2.b)，只是其中一页面来自物联网服务器，另一台来自攻击者的服务器。当我们点击这两个页面上的按钮时，都会向物联网服务器发送一个请求，以设定其温度。我们要将恒温器的温度提高到 99 摄氏度。

图 2: 从三个 URL 获取的 Web 页面

点击第二和第三个页面中的按钮，描述你的观察。哪个可以成功设置恒温器的温度？请解释为什么。如果想找到原因，请在 Firefox 浏览器中点击以下菜单序列，会出现一个窗口显示错误信息 (如果有的话)。提示：出错原因与浏览器实行的同源策略有关。请解释为什么该策略会导致其中一个页面上的操作失败了。

```
Web Developer -> Web Console
```

4.2 任务 2. 攻破同源策略防护

从之前的任务来看，由于浏览器的同源策略的保护机制，看似攻击者页面没法设定恒温器的温度。本任务的目标就是突破这种保护，让我们可以从该页面设置恒温器的温度。

攻破同源策略防护的想法基于这样一个事实：策略的执行是基于主机名，而不是 IP 地址，所以只要我们使用 `www.attacker32.com` 的 URL，就符合同源策略（SOP），但这并不意味着我们只能与 `www.attacker32.com` 这个 Web 服务器进行通信。

在用户浏览器向 `www.attacker32.com` 发送请求之前，它首先需要知道 `www.attacker32.com` 的 IP 地址。所以用户主机会发出一个 DNS 请求，如果 IP 地址不在缓存中，本地 DNS 服务器就会发送 DNS 查询请求到 `www.attacker32.com` 的域名服务器。这个服务器是攻击者的，因此，DNS 的响应是完全受攻击者控制的。

步骤 1: 修改 JavaScript 代码. 在攻击者主机中，`www.attacker32.com/change` 运行的 JavaScript 代码位于文件：`/app/rebind_server/templates/js/change.js` 中。由于该页面来自 `www.attacker32.com` 服务器，根据同源策略，它只能与同一服务器交互。因此我们将代码的第一行从 `http://www.seediot32.com` 改为以下内容 (我们在容器中安装了一个叫做 `nano` 的简单编辑器)：

```
let url_prefix = 'http://www.attacker32.com'
```

完成修改之后，重启攻击者的 Web 服务器容器 (参考下面的命令行)，接着在用户主机上刷新页面，并且重新点击按钮。现在还能看到控制台的错误信息吗？请解释你观察到的现象。

```
$ docker ps
...
78359039627a  attacker-www-10.9.0.180

$ docker container restart 7835
```

步骤 2: 进行 DNS 重绑定. 我们的 JavaScript 代码会发送 HTTP 请求到 `www.attacker32.com`，也就是请求会返回到攻击者的机器，这不是我们想要的，我们想要的是将请求发送到 IoT 服务器。这可以通过 DNS 重绑定技术来实现。在攻击者的域名服务器上，我们先将 `www.attacker32.com` 映射到攻击者 Web 服务器的 IP 地址，这样用户才能从 `http://www.attacker32.com/change` 获得攻击页面。当用户点击网页上的按钮之前，我们改变 `www.attacker32.com` 的映射，这次将其映射到 IoT 服务器的 IP 地址。因此，当用户点击页面的按钮时，触发的请求将到达 IoT 服务器。这正是我们想要的攻击效果。

为了改变 DNS 映射，可以修改攻击者的 DNS 服务器容器内的 `zone_attacker32.com` 文件。该域文件可以在 `/etc/bind` 文件夹中找到。以下是文件的内容。第一项是响应的默认存活时间 `TTL`（单位：秒），即响应在 DNS 缓存中可以保留多长时间。这个值可能需要修改。

```
$TTL 1000
@       IN      SOA   ns.attacker32.com. admin.attacker32.com. (
                2008111001
                8H
                2H
                4W
                1D)

@       IN      NS    ns.attacker32.com.

@       IN      A     10.9.0.22
www     IN      A     10.9.0.22
ns      IN      A     10.9.0.21
*       IN      A     10.9.0.22
```

在对域文件进行更改后，运行以下命令来让域名服务器重新加载修订后的配置。

```
# rndc reload attacker32.com
```

由于之前执行的任务，`www.attacker32.com` 的 DNS 映射已经被本地 DNS 服务器缓存了，直到 1000 秒后才会过期。为了缩短等待时间，我们允许学生使用以下命令清除缓存（在本地 DNS 服务器上）。然而，这个操作只能在攻击开始之前进行。一旦攻击开始，学生不得触碰本地 DNS 服务器。

```
//在本地DNS服务器容器上执行
# rndc flush
```

如果该任务的这两个步骤都正确完成，那么从 `www.attacker32.com` 的页面点击 `change` 按钮，将会使得恒温器的温度设置成功。请在实验报告中给出攻击成功的证据 (截屏)。

4.3 任务 3. 实施攻击

在之前的任务中，用户必须点击按钮来将恒温器设置到一个很高的温度。显然，对于用户来说不太可能去做这样的事。在此任务中，我们创建了一个 Web 页面来自动完成这一操作，你可以通过以下 URL 进行访问该网页：

```
http://www.attacker32.com
```

一旦你在用户主机中加载这一页面，你会看到一个计时器正在从 10 倒数到 0。一旦计时器到达 0，JavaScript 代码会向 `http://www.attacker32.com` 发出一个设置温度的请求，并重新将计时器的计数到 10。学生需要通过使用 DNS 重绑定技术，使得当计时器到达 0 时，温控器的温度会被设置为 88 摄氏度。

5 Submission

你需要提交一个详细的，带有截图的实验报告来描述你做了什么以及你观察到了什么。你还需要对有趣的或是令人惊讶的观察结果进行解释。请同时列出重要的代码段并附上解释。简单地附上代码而不作任何解释将不会得到学分。