Subdomain checking of whitelisted domains could allow unintended redirects

Moderate **JoelSpeed** published **GHSA-4mf2-f3wh-gvf2** on Feb 1, 2021

Package

No package listed

Affected versions

3.0.0 < 6.1.1

Patched versions

> 7.0.0

Description

Impact

What kind of vulnerability is it? Who is impacted?
For users that use the whitelist domain feature, a domain that ended in a similar way to the intended domain could have been allowed as a redirect.

For example, if a whitelist domain was configured for `.example.com`, the intention is that subdomains of `example.com` are allowed.
Instead, `example.com` and `badexample.com` could also match.

Patches

Has the problem been patched? What versions should users upgrade to?
This is fixed in version 7.0.0 onwards.

Workarounds

Is there a way for users to fix or remediate the vulnerability without upgrading?
Disable the whitelist domain feature and run separate OAuth2 Proxy instances for each subdomain.

Original Issue Posted by @semoac:

Whitelist Domain feature is not working as expected because is not matching a dot to ensure the redirect is a subdomain.

Expected Behavior

If whitelist domain is set to `.example.com`, then `hack.alienexample.com` should be rejected as a valid redirect.

Current Behavior

The code is removing the `dot` from `.example.com` and only checking if the redirect string end with `example.com`

Possible Solution

Here

oauth2-proxy/oauthproxy.go
Line 661 in c377466

```
661     if (redirectHostname == domainHostname) || (strings.HasPrefix(domain, ".") && strings.HasSuffix(redirectHostname, domainHostname)) {
```

Include the dot when checking the string:

```
strings.HasSuffix(redirectHostname, "." + domainHostname)
```

Steps to Reproduce (for bugs)

```
package main

import (
        "fmt"
        "strings"
)

func validOptionalPort(port string) bool {
        if port == "" || port == ":*" {
                return true
        }
        if port[0] != ':' {
                return false
        }
        for _, b := range port[1:] {
                if b < '0' || b > '9' {
                        return false
                }
        }
        return true
}

func splitHostPort(hostport string) (host, port string) {
        host = hostport

        colon := strings.LastIndexByte(host, ':')
        if colon != -1 && validOptionalPort(host[colon:]) {
```

```
                host, port = host[:colon], host[colon+1:]
        }

        if strings.HasPrefix(host, "[") && strings.HasSuffix(host, "]") {
                host = host[1 : len(host)-1]
        }

        return
}

func main() {
        domain := ".example.com"
        domainHostname, _ := splitHostPort(strings.TrimLeft(domain, "."))
        redirectHostname := "https://hack.alienexample.com"
        if (strings.HasPrefix(domain, ".") && strings.HasSuffix(redirectHostname, domainHostname)) { fmt.Println("This should not have happen.")}
}
```

Severity

Moderate

CVE ID

CVE-2021-21291

Weaknesses

No CWEs

Credits

 semoac