Vampira

Sevki
s@sevki.org

ABSTRACT

Vampira is a static file server that uses Plan9 namespaces to serve content. It is a very much inspired by the Plan9 HTTP server.

Vampira is a Plan9 HTTP server that uses namespaces to serve static web content.

Vampira much like the original Plan9 **httpd(8)** becomes the none user then it uses the **AddNS** function to contruct a new namespace.

One place vampira doesnt' follow the original plan9 version is that it uses **-map** flags to do rewrites. As vampira is able to bind any arbirtrary path to another what it really has to do is figure out base paths for hosts. In order to do that vampira needs a url to be passed with the aforementioned map flag. This is the RC script I use to run vampira on my server.

```
#!/bin/rc

fn start_vampira {
        rfork
        vampira/vampira -n /n/camdvax/lib/namespace.httpd       -map sevki.io/tmp/vampira/www
}

srv -q tcp!192.168.162.243 camdvax
mount  -q /srv/camdvax /n/camdvax
bind -b /n/camdvax/bin /bin
start_vampira &
```

The **-map**sevki.io/tmp/vampira/www tells vampira that when a request comes in for **sevki.io** the root directory will be **/tmp/vampira/www**

The path rewriting and everything else happens in the namespace file.

```
mount /srv/wiki.sevki /n/sevki.wiki
mount /srv/camdvax /n/camdvax
bind -c /n/camdvax/usr/sevki/blog /tmp/vampira/blog
bind -c /n/camdvax/usr/sevki/www /tmp/vampira/www
bind -c /n/sevki.wiki /tmp/vampira/www/wiki

cd /tmp/vampira
```

In the code above vampira mounts servers, **/srv/wiki.sevki** and **/srv/camdvax** Then files served by those servers are bound to the appropriate places as defined with the **-map** flags.

Then vampira serves everything from the filesystem, cause unlike unix, in plan9 everything implements a file.

```
package main

import (
        "flag"
        "fmt"
        "net/http"
        "net/url"
        "runtime"

        "sevki.org/namespace"
)

type sitemap map[string]string

func (s sitemap) String() string {
        return fmt.Sprint(map[string]string(s))
}

func (s sitemap) Set(value string) error {
        u, err := url.Parse("https://" + value)
        if err != nil {
                panic(err)
        }
        s[u.Host] = u.Path
        return nil
}

func (s sitemap) ServeHTTP(w http.ResponseWriter, r *http.Request) {
        dir := "/tmp/vampira"
        if d, ok := s[r.Host]; ok {
                dir = d
        }
        w.Header().Add("Server-Agent", fmt.Sprintf("vampira/master (%s; %s);", runtime.GOOS, ru
        http.FileServer(http.Dir(dir)).ServeHTTP(w, r)
        return
}

func main() {
        var sitemap = sitemap{}
        nsfile := flag.String("n", "/lib/namespace.httpd", "namespace file")
        addr := flag.String("http", ":3000", "http address to listen to")
        flag.Var(&sitemap, "map", "urls")
        flag.Parse()

        if err := namespace.AddNS(*nsfile, "none"); err != nil {
                fmt.Printf("couldn't build namespace: %v0, err)
                return
        }

        fmt.Printf("Listening on %s...0, *addr)

        if err := http.ListenAndServe(*addr, sitemap); err != nil {
                fmt.Printf("couldn't start httpserver: %v0, err)
                return
        }
}
```