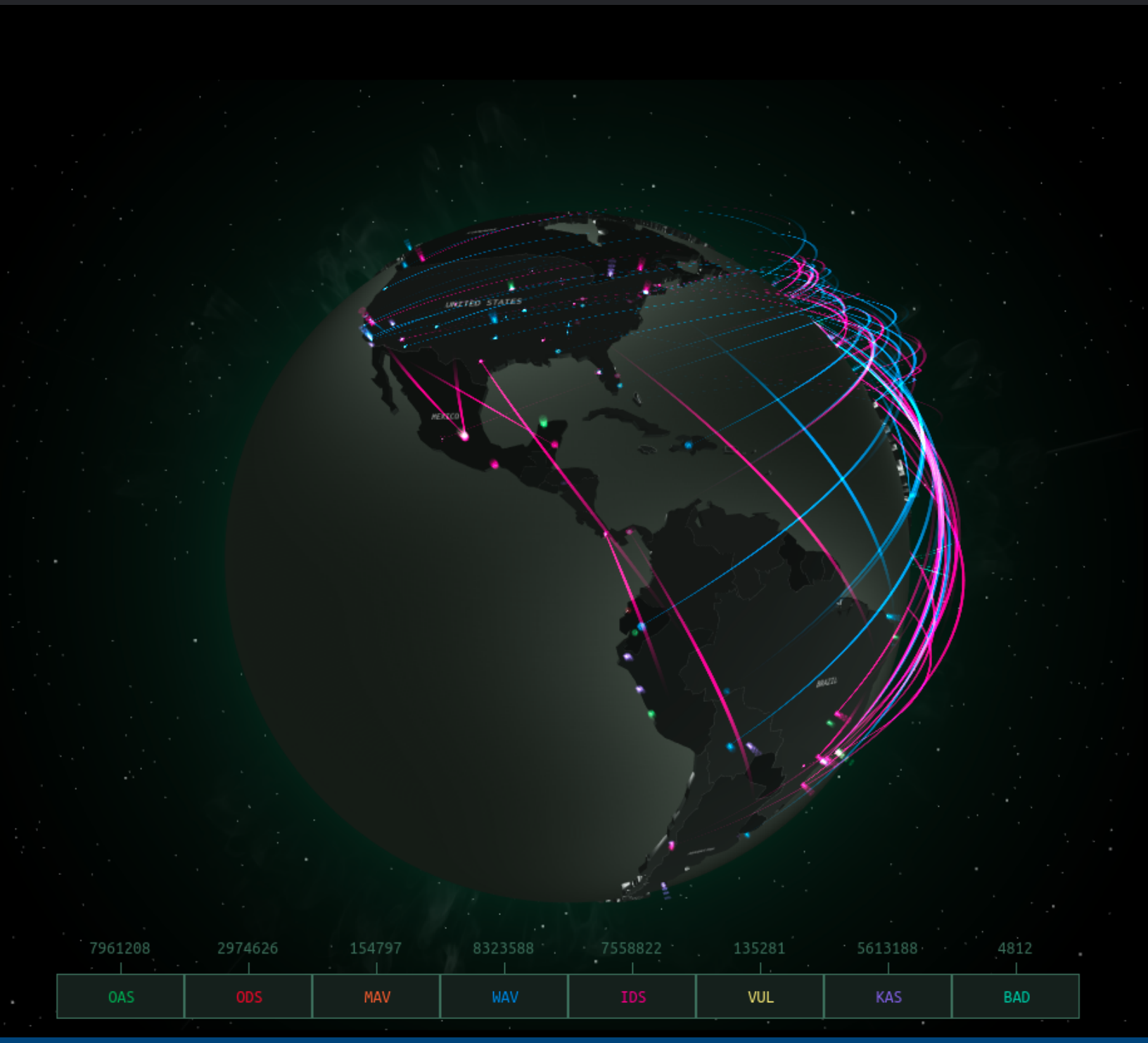
TEC | Tecnológico de Costa Rica

Tarea 2

Mi DDoS

Octubre, 2020



Objetivo

Crear un WebServer el cual utiliza enteramente el protocolo HTTP/1.1 para su comunicación. Este WebServer podría ejecutarse en modo pre-threaded o pre-forked. Además crear un cliente web que busque vencer los webservers dejándolos sin posibilidad de atender a otro cliente.

Datos Generales

- **Fecha de Entrega:** Martes 3 de Noviembre de 2020 antes de las 23:59:59 GMT-6.
- **Fecha de Revisión:** Miércoles 4 de Noviembre de 2020.
- **Lenguaje:** C para GNU/Linux y Python
- **Recurso Humano:** Grupos de 2
- **Valor de la asignación:** 5 %

Profesor

Kevin Moraga
kmoragas@ic-itcr.ac.cr
Escuela de Computación

Introducción

En la actualidad muchos de los webservers se encuentran subutilizados, pero en períodos de tiempo cortos, es posible que se sobre utilicen debido a la demanda del servicio. Esto sucede con muchos proveedores de entradas a eventos masivos. La configuración normalmente no contempla la limitación de los recursos. Existen dos técnicas que pueden ayudar a administrar mejor los recursos de un servidor, ellas son pre-thread y pre-forked.

Requerimientos

Requerimientos Funcionales

Pre-thread WebServer Se debe de crear un web server el cual implemente la técnica llamada prethread. Esta técnica consiste en crear previamente varios hilos de ejecución del método que atiende la solicitudes. Estos **hilos** se crean utilizando la biblioteca pthreads de Unix. Debe de recibir como parámetro el número de **hilos** N que se deben pre-crear, el webserver escuchará en el puerto estándar de HTTP, y tendrá N hilos posibles para atender la solicitud, cada solicitud se atenderá por el primer hilo que esté disponible. En caso que no existan más disponibles mostrará un mensaje de error, indicando que se ha sobrepasado el número de clientes que pueden ser atendidos.

Pre-forked WebServer Se debe de crear un web server el cual implemente la técnica llamada **pre-forked**. Esta técnica consiste en crear previamente varios **procesos** del método que atiende la solicitudes. Estos **procesos** se crean utilizando el **system call** estándar de Unix. Debe de recibir como parámetro el número de **procesos** N que se deben pre-crear, el webserver escuchará en el puerto estándar de HTTP, y tendrá N procesos posibles para atender la solicitud, cada solicitud se atenderá por el primer proceso que esté disponible. En caso que no existan más disponibles mostrará un mensaje de error, indicando que se ha sobrepasado el número de clientes que pueden ser atendidos.

HTTPClient Se debe crear un cliente HTTP el cual permita descargar un recurso. Para ello utilice la biblioteca curl en el lenguaje Python.

HTTPClient en C Se debe crear un cliente HTTP el cual permita descargar un recurso. Para ello utilice el lenguaje de programación C.

StressCMD Se debe crear una aplicación que reciba como parámetro un ejecutable (Además de los parámetros del ejecutable). Luego debe de crear la cantidad de hilos que el cliente especifique, con el objetivo de lanzar un ataque de Denegación de Servicio. El principal objetivo de unir el HTTPClient y el StressCMD es de saturar los webservers hasta que estos se queden posibilidad de atender otro cliente más. En el lenguaje Python.

CGI Injector (exploit) Este cliente se encargará de inyectar código nuevo al servidor anteriormente creado.

Además de las definiciones anteriores tome en cuenta:

- El WebServer deberánde servir los archivos que se encuentren disponibles en la dirección especificada en los parámetros.
- El WebServer debe de implementar los métodos PUT, GET, POST y DELETE de HTTP Y debe permitir la ejecución de binarios escritos en C. Emulando la funcionalidad de los CGI's.
- Es necesario documentar todos los métodos de HTTP incluyéndolos en la introducción de la documentación.
- El WebServer debe de permitir cambiar de protocolo, y poder implementar una respuesta válida para una consulta con cualquiera de los siguientes protocolos: FTP, SSH, SMTP, DNS, TELNET, SNMP. Para utilizar los protocolos se puede incluir un parámetro más, al momento de instanciar el servidor web, que defina el protocolo. O bien, utilizar el protocolo, descriminándolo a partir del puerto por defecto.

Requerimientos Técnicos

- El desarrollo se debe de realizar utilizando el lenguaje de programación C para GNU/Linux.
- El desarrollo se debe de realizar utilizando el lenguaje de programación C.
- Es necesario utilizar la biblioteca pthreads de C para el pre-trhead Server.
- El servidor deben de funcionar utilizando GNU/Linux

La sintaxis del web server prethread será:

```
$ prethread-webserver -n <cantidad-hilos> -w <path-www-root> -p <port>
```

La sintaxis del web server preforked será:

```
$ preforked-webserver -n <cantidad-hilos> -w <path-www-root> -p <port>
```

La sintaxis del cliente web:

```
$ httpclient -u <url-de-recurso-a-obtener>
```

La sintaxis del stress:

```
$ stress -n <cantidad-hilos> httpclient <parametros del cliente>
```

Nótese que el cliente debe de recibir los parámetros desde la terminal. El signo de dolar representa el shell del SO.

Aspectos Administrativos

Entregables

- Código fuente del programa que cumpla los requerimientos funcionales y técnicos.
- Binario del programa, compilado para una arquitectura x86.
- Fuente de la documentación en MD o Latex y luego a PDF.
- PDF con la documentación.

Evaluación

- WebServer: 40 %
  - Implementación del Pre-thread
  - Implementación del Pre-forked
  - Servir archivos grandes correctamente
  - Desplegar una página web completa desde un navegador
  - Implementación del un CGI
- Implementación de Protocolos: 10 %
- http-Client: 5 %
- http-Client en C: 5 %
- Stress-Client: 10 %
- CGI Injector (Exploit): 10 %
- Documentación utilizando MD: 20 %

Documentación

Las siguientes son las instrucciones para la documentación. NO LA IMPRIMA. Además la documentación se debe de realizar utilizando MD con Latex.

1. **Introducción**: Debe presentar el problema, utilizando una redacción propia, discutido y redactado en el kick-off de la asignación.
2. **Ambiente de desarrollo**: Indicar las herramientas usadas para implementar la tarea.
3. **Estructuras de datos usadas y funciones**: Se debe describir las principales funciones y estructuras utilizadas en la elaboración de esta asignación.
4. **Instrucciones para ejecutar el programa**: Presentar las consultas concretas usadas para correr el programa para el problema planteado en el enunciado de la tarea y para los casos planteados al final de esta documentación.
5. **Actividades realizadas por estudiante**: Este es un resumen de las bitácoras de cada estudiante (estilo timesheet) en términos del tiempo invertido para una actividad específica que impactó directamente el desarrollo del trabajo, de manera breve (no más de una línea) se describe lo que se realizó, la cantidad de horas invertidas y la fecha en la que se realizó. Se deben sumar las horas invertidas por cada estudiante, sean concientes a la hora de realizar esto el profesor determinará si los reportes están acordes al producto entregado.
6. **Autoevaluación**: Indicar el estado final en que quedó el programa, problemas encontrados y limitaciones adicionales. Adicionalmente debe de incluir el reporte de commits de git. Por otro lado, también debe incluir una calificación con la rúbrica de la sección "Evaluación".

7. **Lecciones Aprendidas** del proyecto: Orientados a un estudiante que curse el presente curso en un futuro.
8. **Bibliografía** utilizada en la elaboración de la presente asignación.
9. Es necesario documentar el código fuente.

Aspectos Adicionales

Aún cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones:

1. Sí no se entrega documentación, automáticamente se obtiene una nota de 0.
2. Sí el código no compila se obtendrá una nota de 0, por lo cuál se recomienda realizar la defensa con un código funcional.
3. El código debe ser desarrollado en el lenguaje especificado en los Datos Generales, en caso contrario se obtendrá una nota de 0.
4. Sí no se siguen las reglas del formato del envío a través de Google Drive se obtendrá una nota de 0.
5. La revisión de la documentación será realizada por parte del profesor, no durante la defensa del proyecto.
6. Cada grupo tendrá como máximo 20 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cuál se recomienda tener todo listo antes de ingresar a la defensa.
7. Cada excepción o error que salga durante la ejecución del proyecto y que se considere debió haber sido contemplada durante el desarrollo del proyecto, se castigará con 2 puntos de la nota final de la presente asignación.
8. Cada grupo es responsable de llevar los equipos requeridos para la revisión, si no cuentan con estos deberá avisar al menos 2 días antes de la revisión al profesor para coordinar el préstamo de estos.
9. Durante la revisión podrán participar asistentes, otros profesores y el coordinador del área.
10. Cualquier indicio de copia será calificado con una nota de 0 y será procesado de acuerdo al reglamento.