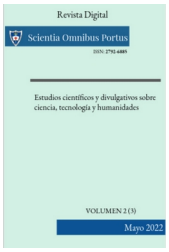
Scientia Omnibus Portus
ISSN: 2792-6885

https://iescelia.org/ojs



Artículo

Acercamiento a la Ciberseguridad con NIDS

Eduardo Tapia
IES Celia Viñas
etappad369@g.educaand.es

Resumen.-

El artículo elaborado responde a la necesidad de formar a los/as profesionales de la Ciencia de la Computación en aspectos relacionados con la seguridad de los sistemas informáticas.- Actualmente, se concibe la seguridad de un sistema informático como un todo .- Para ello, es necesario el compromiso de los Responsables de la Organización de Recursos que ayuden a determinar las prioridades y recursos que se van a destinar a la protección de los Activos de Información .- Esta visión conjunta, compacta e integrada de la seguridad de los Activos de Información es relativamente reciente , lo que ha supuesto un reto para la elaboración de este artículo, que conjuga conocimientos recientes con desarrollos intelectuales del inicio de la Ciencia de la Computación .- Esta mezcla de razonamientos pone de manifiesto lo fascinante que resulta la investigación para descubrir soluciones a los problemas que la Humanidad encuentra en su desarrollo.-

Palabras clave: *artículo, científico, divulgación, publicación, manuscrito, IDS, NIDS.-*

Recibido el 15/03/2022
Aceptado el 16/04/2022

Introducción.-

Vamos a descubrir como funciona una parte de las herramientas que actualmente utilizamos para proteger los Activos de Información.- Espero que resulte interesante, si no apasionante, la utilización de herramientas que, actualmente, aseguran el funcionamiento de los Sistemas Informáticos que dan forma a nuestra sociedad actual. Agradecer su tiempo; espero que sea de su agrado.-

Sintáxis de las Reglas de NIDS Snort .-

La reglas de NIDS **Snort** las podemos dividir en dos secciones lógicas : ***cabecera de la regla y opciones***:

- La **cabecera** contiene *la acción de la regla en sí, protocolo, IPs, máscaras de red, puertos origen y destino y destino del paquete o dirección de la operación.*
    - *Acción*
    - *Protocolos involucrados*
    - *Direcciones IP*
    - *Números de puerto*
    - *Dirección de la operación*

- La sección **opciones** contiene *los mensajes y la información necesaria para la decisión a tomar por parte de la alerta en forma de opciones.*
    - *Mensaje*
    - *Opciones de decisión*

Ejemplo de Reglas de NIDS Snort.-

Alerta en el caso de que el paquete de información transferido entre ordenadores contenga la palabra ALERTA.

Ejemplo de Regla para Snort:

alert ip any any -> any any (sid:1;msg:»Word ALERTA found»;content:»ALERTA»;)

Veamos que significa la sintáxis de esta regla :

- alert: Permite activar una alerta si la regla coincide .-
- ip: Compara las reglas con cualquier protocolo, a saber , TCP, UDP o ICMP .-
- any any > any any: Indica que analizará cualquier host de origen y puerto a cualquier host y puerto de destino .
- sid:1;msg:»Word ALERTA found»: Identificador de la regla y el mensaje que se enviará con la alerta .-

El contenido de la opción es la particularidad de esta regla. Siempre que se realiza una coincidencia de patrón de opción de contenido, se llama a la función de coincidencia de patrón de Boyer-Moore y la prueba (muy costosa desde el punto de vista computacional) se realiza contrastando el contenido del paquete .

Implementación Algoritmo de búsqueda de cadenas Boyer-Moore en Lenguaje de Programación C .-

```
# include <limits.h>
# include <string.h>

# define ALPHABET_SIZE (1 << CHAR_BIT)

static void compute_prefix(const char* str, size_t size, int result[size]) {
 size_t q;
 int k;
 result[0] = 0;

 k = 0;
 for (q = 1; q < size; q++) {
     while (k > 0 && str[k] != str[q])
         k = result[k-1];

     if (str[k] == str[q])
         k++;
     result[q] = k;
 }
}
```

```
static void prepare_badcharacter_heuristic(const char *str, size_t size,
 int result[ALPHABET_SIZE]) {

 size_t i;

 for (i = 0; i < ALPHABET_SIZE; i++)
     result[i] = -1;

 for (i = 0; i < size; i++)
     result[(size_t) str[i]] = i;
}

void prepare_goodsuffix_heuristic(const char *normal, size_t size,
 int result[size + 1]) {

 char *left = (char *) normal;
 char *right = left + size;
 char reversed[size+1];
 char *tmp = reversed + size;
 size_t i;

 /* reverse string */
 *tmp = 0;
 while (left < right)
     *(--tmp) = *(left++);

 int prefix_normal[size];
 int prefix_reversed[size];

 compute_prefix(normal, size, prefix_normal);
 compute_prefix(reversed, size, prefix_reversed);

 for (i = 0; i <= size; i++) {
     result[i] = size - prefix_normal[size-1];
 }

 for (i = 0; i < size; i++) {
     const int j = size - prefix_reversed[i];
     const int k = i - prefix_reversed[i]+1;

     if (result[j] > k)
         result[j] = k;
 }
}
/*
 * Algoritmo de Búsqueda de Boyer-Moore
 */
const char *boyermoore_search(const char *haystack, const char *needle) {
 /*
 * Calc string sizes
 */
 size_t needle_len, haystack_len;
 needle_len = strlen(needle);
 haystack_len = strlen(haystack);

 /*
 * checkeo
 */
 if(haystack_len == 0)
     return NULL;
 if(needle_len == 0)
     return NULL;
 if(needle_len > haystack_len)
     return NULL;
```

```
 /*
 * Inicio de la Heurística
 */
 int badcharacter[ALPHABET_SIZE];
 int goodsuffix[needle_len+1];

 prepare_badcharacter_heuristic(needle, needle_len, badcharacter);
 prepare_goodsuffix_heuristic(needle, needle_len, goodsuffix);

 /*
 * Búsqueda Boyer-Moore
 */
 size_t s = 0;
 while(s <= (haystack_len - needle_len))
 {
     size_t j = needle_len;
     while(j > 0 && needle[j-1] == haystack[s+j-1])
         j--;

     if(j > 0)
     {
         int k = badcharacter[(size_t) haystack[s+j-1]];
         int m;
         if(k < (int)j && (m = j-k-1) > goodsuffix[j])
             s+= m;
         else
             s+= goodsuffix[j];
     }
     else
     {
         return haystack + s;
     }
 }

 /* not found */
 return NULL;
}
```

Variantes : Algoritmo Boyer - Moore Turbo.-

Toma un espacio adicional para completar la búsqueda en 2n comparaciones y no en 3n comparaciones realizadas por el algoritmo original, donde n es el número de caracteres en el texto donde ha de ser buscada la palabra.

Conclusiones .-

Las soluciones de base de la Ciencia de la Computación parten de muy atrás, del inicio de la misma, actualizadas constantemente y adaptadas a las herramientas actuales .- Supone un reconocimiento al esfuerzo de los/as científicos/as que le dieron forma a esta Ciencia.-

En base a estas investigaciones y su desarrollo posterior, disponemos de una serie de herramientas y estratégias de diseño de mecanismos de protección de los Activos Automatizados, tales como datos médicos, financieros, educativos, suministros básicos , etc...

Referencias Bibliográficas .-

[1]R. S. Boyer; Strother Moore, J.. (1977). A fast string searching algorithm. USA; Comm. ACM.

[2]Kerry J. Cox, Christopher Gerg. O'reilly Associates. (2004). Snort and IDS Tools. Sebastopol (EE.UU); O'reilly associates.

Biografía autor.-

Eduardo Tapia .- Profesor de Enseñanza Secundaria en la especialidad de Informática desde hace 20 años (pff! Como pasa el tiempo!).- Paisano de Almería, apasionado de sus playas y urbanita aficionado, me encanta la enseñanza y el reto que supone .-