Formula extemporánea

Este método consiste en obtener e indexar data procedente del procesador principal u otros componentes del hardware para generar un array de datos, de tal forma que al segmentar el mismo y aplicando método diferencial se pueda de manera indirecta conocer si estamos expuestos a un proceso o dispositivo malintencionado.

Dado que la máxima velocidad de calculo alcanzable en nuestro host esta determinada por el procesador y los componentes asociados al mismo, cualquier evento o proceso que interactúe con este terminara por alterar de manera notable la correlación de datos que nuestra app genera durante los distintos procesos que ejecute.

A continuación se expone la formula para arreglar dicha data, siendo esta (formula base) de indexación y pudiéndose agregar más parámetros según el nivel de seguridad requerido. El array de datos o segmentos derivados de este proceso es nuestra (llave) para generar el motor de cifrado. Una vez concluido este proceso dicho motor solo es valido para un tipo de fractal o arreglo de datos. Aun cuando ejecutemos procesos paralelos y la data sea la misma la resultante sera diferente. Esto a su vez sirve como identificador único de autenticidad e integridad de datos.

Para este ejemplo nuestros parámetros a considerar son:

Vcore: voltaje del procesador
clk: Speed clock (velocidad del procesador en Hz)
Rtc: tiempo del sistema

Con lo anterior vamos a conformar nuestra referencia temporal (Trf).

Trf = [n1,n2,n3,n4] + [cl1,cl2,cl3,cl4] + [v1,v2,v3,v4]

donde:

(n1 → n4) = los últimos 4 dígitos de nuestro (Rtc) expresados en milisegundos.
(cl1 → cl4) = los últimos 4 dígitos de nuestro (clk) expresados en Hercios.
(v1 → v4) = los últimos 4 dígitos del voltaje flotante del procesador expresados en milivolts.

Los últimos 4 dígitos corresponde a los más a la derecha, siendo estos la máxima (rms) posible de adquirir por nuestro sistema.

Comenzamos:

Trf1: a = (cl4 + n4); b = (cl3 + n3); c = (cl2 + n2); d = (cl1 + n1)

nuestra formula (envenenada) queda:

ejemplo (1):

a1, b1 → (Ti)1 → (Ti)1, c1) → d1
c1 ←---------- (fx)
c0 ----------→ ↑

donde:
a1 = (cl4); b1 = (n4); (Ti)1 = (a1 + b1); c0 = (v4); (fx) = (kpn → a1); c1 = (c0 + fx); d1 = (Ti)1 + c1)

Calc:

a1 = 7; b1 = 3; c0 = 5 [7,3 → (1) → (1,7) → (8)]

a1 = 6; b1 = 3; c0 = 5 [6,3 → (9) → (9,8) → (8)]

a1 = 7; b1 = 5; c0 = 5 [7,5 → (3) → (3,7) → (1)]

a1 = 7; b1 = 3; c0 = 5 [7,3 → (1) → (1,7) → (8)]

a1 = 7; b1 = 3; c0 = 8 [7,3 → (1) → (1,1) → (2)]

a1 = 7; b1 = 3; c0 = 3 [7,3 → (1) → (1,5) → (6)]

ejemplo (2):

a1, b1 → (Ti)1 → (Ti)1, c1) → d1
c1 ←---------- (fx)
c0 ----------→ ↑

donde:

a1 = (cl4); b1 = (n4); (Ti)1 = (a1 + b1); c0 = (kpn → b1); (fx) = v4; d1 = (Ti)1 + c1)

Calc:

a1 = 7; b1 = 3; c0 = 6; v4 = 5 [7,3 → (1) → (1,2) → (3)]

a1 = 6; b1 = 3; c0 = 6; v4 = 5 [6,3 → (9) → (9,2) → (2)]

a1 = 7; b1 = 5; c0 = 4; v4 = 5 [7,5 → (3) → (3,9) → (3)]

a1 = 7; b1 = 3; c0 = 6; v4 = 8 [7,3 → (1) → (1,5) → (6)]

En el ejemplo (1) a1 transfiere su (Kpn) a (fx) para de esta forma modificar (c0) y obtener (c1)
para el ejemplo (2) la (Kpn → b1) genera el parámetro (c0) el cual es modificado por (v4), es decir muestra (fx). La (fx) es el factor que utilizamos para decirle a un valor que se comporte como otro. Esto le llamamos envenenar la ecuación y es de gran utilidad para generar cifrados dinámicos.
Esto es así porque para efectos de un auditor el cuerpo del programa se observa de la forma correcta pero nosotros generamos un punto de entrada [EP] en el cual nuestro programa una vez se cumpla nuestra condicional toma un valor determinado y lo calcula utilizando nuestra (fx) de esta forma se genera un programa o data paralela dentro del cuerpo del mismo. Este no es visible a menos que se posea los parámetros para verificar el [PE]. Intentar hallar este valor por fuerza bruta es casi imposible, ya que como mostramos en los ejemplos anteriores para diferentes factores obtenemos mismo producto. Entonces nuestra ecuación es fácil de verificar en sentido de avance (calculo) pero es ambigua en reversa, a menos que se posean los valores de arranque.

Esta ecuación difiere de las de (compresión) caracterización de datos ya que en este caso se requiere lo contrario. Es decir para caracterizar datos nuestra posibilidad de colisión y o ambigüedad debe de ser (0) de lo contrario obtendríamos datos sin coherencia. Los detalles de este ultimo caso se omite por razones de copyright y aprovechamiento comercial.

Entonces:

la resultante de estas 4 operaciones lógicas conforman nuestro wachtdog [WD]

(Trf4, Trf3, Trf2, Trf1) = WD1

la [DIFF]→] entre [WD]1] + [WD]2] debe de ser siempre > que [WD]3] y este ciclo se repetite desplazando el valor a calcular utilizando siempre los últimos 3 [WD].

Lo anterior nos genera un plot que le permite a nuestro programa predecir o confirmar cualquier operación anormal en lo referente a depuración del proceso o esnifado con fines de ingeniería inversa o malintencionados para con nuestra app.

Dado que los parámetros anteriores se toman en la placa base a bajo nivel, cualquier interacción con nuestra app genera una ruptura de correlación en nuestro plot, pero permite que el sistema opere de manera normal para con los demás (Pid’s). Entonces mientras más esfuerzo (interacción o calculo) realice una aplicación para con nuestro programa. Más rápido se cumple nuestra condición (halt) [HLT] derivando en la implementación de contramedias (varias).

En una ecuación de confirmación de valor tenemos:

a,b → (Ti)1 → (Ti)1,c) = d
  c

Ejemplo:

justificamos (confirmamos) valor (b)

nuestros factores serian (b,c); a = (Kpn → c); d = nuestro resultado.

b = 3; c = 7 [2,3 → (5) →(5,7) = (3)

b = 8; c = 7 [2,8 → (1) →(1,7) = (8)

b = 2; c = 8 [1,2 → (3) →(3,8) = (2)

b = 9; c = 1 [8,9 → (8) →(8,1) = (9)

justificamos (confirmamos) valor (c)

nuestros factores serian (b,c); a = (Kpn → b); d = nuestro resultado.

b = 3; c = 7 [6,3 → (9) →(9,7) = (7)

b = 8; c = 7 [1,8 → (9) →(9,7) = (7)

b = 2; c = 8 [7,2 → (9) →(9,8) = (8)

b = 9; c = 1 [9,9→ (9) →(9,1) = (1)

entonces:

podemos afirmar que el proceso de envenenar una ecuación consiste en modificar la relación entre el valor esperado por (b) en la posición (a) y la Kpn de (b), (c) según sea el caso.

Existen varios modelos adulterados (envenenados) siendo este el modelo básico de dos componente o factores primarios, pero se pueden indexar tantos como se requieran para conformar array complejos. Así como cruzar (intercambiar) valores entre estos de manera dinámica haciendo más robusta nuestra data final en lo referente a reversión.

Julio C. Gómez
io-exchange.com (c)