Verificación independiente de la funcionalidad de IPsec en FreeBSD

David Honig <honig@sprynet.com>

Revision: 52902

2019-03-29 15:17:05 por gabor.

Resumen

Instaló IPsec y parece estar funcionando. ¿Cómo lo sabe? Describo un método para verificar de forma experimental que IPsec está funcionando.

Tabla de contenidos

1. El problema

Primero, asumamos que ha instalado *IPsec*. ¿Cómo sabe que está funcionando? Claro, su conexión no funcionará si está mal configurada, y funcionará cuando finalmente lo haga bien. netstat(1) la listará. ¿Pero puede confirmarlo de forma independiente?

2. La solución

Primero, alguna información teórica relevante sobre criptografía:

1. Los datos cifrados se distribuyen uniformemente, es decir, tienen una entropía máxima por símbolo;

2. Los datos sin procesar y sin comprimir suelen ser redundantes, es decir, tienen una entropía submáxima.

Suponga que usted pudiera medir la entropía de los datos que van hacia -y desde- su interfaz de red. Entonces podría ver la diferencia entre los datos no cifrados y los cifrados. Esto sería verdad incluso si algunos de los datos

en “modo cifrado” no lo estuvieran---ya que el encabezado IP más externo debe estarlo para que el paquete sea enrutable.

2.1. MUST

El “Universal Statistical Test for Random Bit Generators” (MUST) de Ueli Maurer mide rápidamente la entropía de una muestra. Utiliza un algoritmo de compresión. El código se proporciona a continuación para una variante que mide partes sucesivas (~cuarto de megabyte) de un archivo

2.2. Tcpdump

También necesitamos una forma de capturar los datos de red sin procesar. Un programa llamado tcpdump(1) le permite hacerlo, si tiene habilitada la interfaz de *Berkeley Packet Filter* en el archivo de configuración de su kernel.

El comando:

```
tcpdump -c 4000 -s 10000 -w  dumpfile.bin
```

capturará 4000 paquetes sin procesar en el fichero *`dumpfile.bin`*. En este ejemplo se capturarán hasta 10.000 bytes por paquete.

3. El experimento

Aquí está el experimento:

1. Abra una ventana a un host IPsec y otra ventana a un host inseguro.

2. Ahora empiece a capturar paquetes.

3. En la ventana “segura”, ejecute el comando UNIX® yes(1), que transmitirá el carácter `y`. Después de un rato, detenga el comando. Cambie a la ventana insegura, y repita. Espere un poco, detenga el comando.

4. Ahora ejecute MUST en los paquetes capturados. Debería ver algo como lo siguiente. Lo importante a tener en cuenta es que la conexión segura tiene un 93% (6,7) del valor esperado (7,18), y la conexión “normal” tiene un 29% (2,1) del valor esperado.

```
% tcpdump -c 4000 -s 10000 -w  ipsecdemo.bin
% uliscan ipsecdemo.bin

Uliscan 21 Dec 98
L=8 256 258560
Measuring file ipsecdemo.bin
Init done
Expected value for L=8 is 7.1836656
6.9396 ------------------------------------------------------
6.6177 ----------------------------------------------------
6.4100 --------------------------------------------------
2.1101 -----------------
2.0838 -----------------
2.0983 -----------------
```

4. Advertencia

Este experimento muestra que IPsec *parece* estar distribuyendo los datos de la carga útil *uniformemente*, como debe hacerlo el cifrado. Sin embargo, el experimento aquí descrito *puede no* detectar muchas de las posibles fallas del sistema (para las cuales no tengo evidencias). Esto incluye la generación o intercambio de claves deficientes, datos o claves visibles para otros, uso de algoritmos débiles, subversión del kernel, etc. Estudie el código; conozca el código.

5. IPsec---Definición

Extensiones de seguridad del Protocolo de Internet para IPv4; requerido para IPv6. Un protocolo para negociar el cifrado y la autenticación a nivel de IP (host a host). SSL solo protege un socket de aplicación. SSH protege solo el login. PGP protege un archivo o mensaje específico. IPsec encripta todo entre dos hosts.

6. Instalando IPsec

La mayoría de las versiones modernas de FreeBSD soportan IPsec en su código base. Por lo tanto, deberá incluir la opción `IPSEC` en la configuración de su kernel y, después de recompilar y reinstalar el kernel, configure las conexiones de IPsec usando el comando setkey(8).

En el Manual de FreeBSD se proporciona una guía completa sobre cómo ejecutar IPsec en FreeBSD.

7. src/sys/i386/conf/KERNELNAME

Esto debe estar presente en el archivo de configuración del kernel para capturar datos de red con tcpdump(1). Asegúrese de ejecutar config(8) después de agregar esto, recompilar y reinstalar.

```
device bpf
```

8. Maurer's Universal Statistical Test (tamaño de bloque=8 bits)

Puede encontrar el mismo código fuente en este enlace.

```
/*
  ULISCAN.c   ---blocksize of 8

  1 Oct 98
  1 Dec 98
  21 Dec 98       uliscan.c derived from ueli8.c

  This version has // comments removed for Sun cc

  This implements Ueli M Maurer's "Universal Statistical Test for Random
  Bit Generators" using L=8

  Accepts a filename on the command line; writes its results, with other
  info, to stdout.

  Handles input file exhaustion gracefully.

  Ref: J. Cryptology v 5 no 2, 1992 pp 89-105
  also on the web somewhere, which is where I found it.

  -David Honig
  honig@sprynet.com

  Usage:
  ULISCAN filename
  outputs to stdout
*/

#define L 8
#define V (1<<L)
#define Q (10*V)
#define K (100   *Q)
#define MAXSAMP (Q + K)
```

```
#include <stdio.h>
#include <math.h>

int main(argc, argv)
int argc;
char **argv;
{
  FILE *fptr;
  int i,j;
  int b, c;
  int table[V];
  double sum = 0.0;
  int iproduct = 1;
  int run;

  extern double   log(/* double x */);

  printf("Uliscan 21 Dec 98 \nL=%d %d %d \n", L, V, MAXSAMP);

  if (argc < 2) {
    printf("Usage: Uliscan filename\n");
    exit(-1);
  } else {
    printf("Measuring file %s\n", argv[1]);
  }

  fptr = fopen(argv[1],"rb");

  if (fptr == NULL) {
    printf("Can't find %s\n", argv[1]);
    exit(-1);
  }

  for (i = 0; i < V; i++) {
    table[i] = 0;
  }

  for (i = 0; i < Q; i++) {
    b = fgetc(fptr);
    table[b] = i;
  }

  printf("Init done\n");

  printf("Expected value for L=8 is 7.1836656\n");

  run = 1;

  while (run) {
    sum = 0.0;
    iproduct = 1;

    if (run)
      for (i = Q; run && i < Q + K; i++) {
        j = i;
        b = fgetc(fptr);

        if (b < 0)
          run = 0;

        if (run) {
          if (table[b] > j)
            j += K;

          sum += log((double)(j-table[b]));
```

```
          table[b] = i;
        }
      }

    if (!run)
      printf("Premature end of file; read %d blocks.\n", i - Q);

    sum = (sum/((double)(i - Q))) /  log(2.0);
    printf("%4.4f ", sum);

    for (i = 0; i < (int)(sum*8.0 + 0.50); i++)
      printf("-");

    printf("\n");

    /* refill initial table */
    if (0) {
      for (i = 0; i < Q; i++) {
        b = fgetc(fptr);
        if (b < 0) {
          run = 0;
        } else {
          table[b] = i;
        }
      }
    }
  }
}
```