Verificação Independente da Funcionalidade IPsec no FreeBSD

David Honig <honig@sprynet.com >
Revisão: 52676

2018-12-12 12:55:19 por ebrandi.

Resumo

Você instalou o IPsec e ele parece estar funcionando. Como você sabe? Eu descrevo um método para verificar experimentalmente se o IPsec está funcionando.

Índice

1. O problema

Primeiro, vamos assumir que você tem o *IPsec* instalado . Como você sabe que ele está funcionando? Claro, sua conexão não funcionará se ele estiver mal configurado, e funcionará quando você finalmente acertar a configuração. O netstat(1)irá listá-lo. Mas você pode confirmar isso de forma independente?

2. A solução

Em primeiro lugar, vejamos alguma informação teórica relevante em relação à criptografia:

1. Dados criptografados são uniformemente distribuídos, ou seja, possuem entropia máxima por símbolo;

2. Os dados brutos, não comprimidos são tipicamente redundantes, isto é, possuem entropia submáxima.

Suponha que você possa medir a entropia dos dados destinados para a sua interface de rede e também dos dados originados dela. Então você pode ver a diferença entre dados não criptografados e dados criptografados. Isso seria

verdade mesmo que alguns dos dados no “modo criptografado” não estivessem criptografados --- como deve o cabeçalho IP mais externo para que o pacote seja roteável.

2.1. MUST

O teste de “Estatística Universal para Geradores de Bits Aleatórios” de Ueli Maurer (MUST) mede rapidamente a entropia de uma amostra. Ele usa um algoritmo semelhante à compressão. O código é dado abaixo para uma variante que mede partes sucessivas (aproximadamente um quarto de megabyte) de um arquivo.

2.2. Tcpdump

Também precisamos de uma maneira de capturar os dados brutos da rede. Um programa chamado tcpdump(1) permite que você faça isso, se você ativou a interface *Berkeley Packet Filter* no seu arquivo de configuração do kernel.

O comando:

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

irá capturar 4000 pacotes brutos no arquivo *`dumpfile.bin`* . Até 10.000 bytes por pacote serão capturados neste exemplo.

3. O Experimento

Aqui está o experimento:

1. Abra uma janela para um host IPsec e outra janela para um host inseguro.

2. Agora comece a capturar os pacotes.

3. Na janela “segura”, execute o comando UNIX® yes(1), que transmitirá o caractere `y`. Depois de um tempo, pare com isso. Alterne para a janela insegura e repita. Depois de um tempo, pare.

4. Agora execute o MUST nos pacotes capturados. Você deve ver algo como o seguinte. O importante é notar que a conexão segura tem 93% (6,7) do valor esperado (7,18), e a conexão “normal” tem 29% (2,1) do 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. Embargo

Esta experiência mostra que o IPsec *parece* estar distribuindo os dados de carga *uniformemente*, como a criptografia deveria. No entanto, o experimento descrito aqui *não pode* detectar muitas das falhas possíveis em um sistema (nenhum dos quais eu tenho qualquer evidência para). Estes incluem geração ou troca deficiente de chaves, dados ou chaves sendo visíveis para outros, uso de algoritmos fracos, subversão do kernel, etc. Estude a fonte; conheça o código.

5. IPsec --- Definição

Extensões de segurança do protocolo Internet para o IPv4; obrigatório para o IPv6. Um protocolo para negociar criptografia e autenticação no nível IP (host para host). O SSL protege apenas um soquete de aplicativo; O SSH protege apenas um login;PGP protege apenas um arquivo ou mensagem especifico. O IPsec criptografa tudo entre dois hosts.

6. Instalando o IPsec

A maioria das versões modernas do FreeBSD tem suporte a IPsec em sua fonte base. Portanto, você precisará incluir a opção `IPSEC` em sua configuração de kernel e, após a reconstrução e reinstalação do kernel, configurar as conexões IPsec usando o comando setkey(8).

Um guia completo sobre como executar o IPsec no FreeBSD é fornecido no Handbook do FreeBSD.

7. src/sys/i386/conf/KERNELNAME

Isto precisa estar presente no arquivo de configuração do kernel para habilitar o suporte para captura de dados de rede com o tcpdump(1). Certifique-se de executar o config(8) depois de adicionar a linha, de recompilar e de reinstalar.

```
device bpf
```

8. Teste Estatístico Universal de Maurer (para tamanho de bloco = 8 bits)

Você pode encontrar o mesmo código em neste link.

```
/*
  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;
        }
      }
    }
  }
}
```