Module 17
Buffer Overflow

Các Chủ Đề Chính Trong Chương Này

Tổng Quan Về Buffer Overflow

Shell Code

khai thác buffer overflow

Tìm Kiếm Lỗi Tràn Bộ Đệm

Minh Họa Khai Thác Lỗi Tràn Bộ Đệm

Tổng Quan Về Buffer Overflow

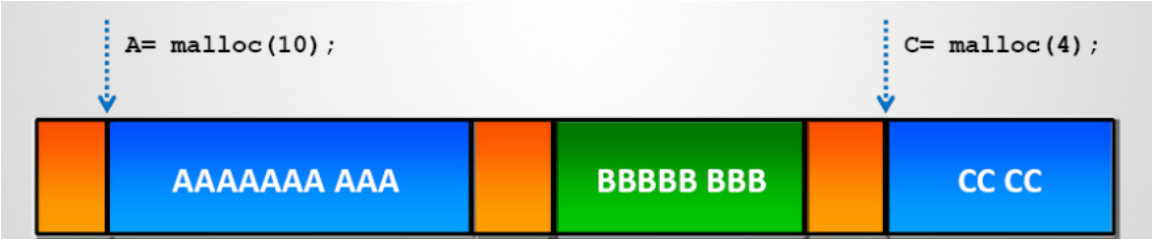
Buffer Overflow hay **BoF** là lỗi tràn bộ đệm, có nguyên nhân gần giống với tình huốn tấn công SQL injection khi người dùng hay hacker cung cấp các biến đầu vào hay dữ liệu vượt quá khả năng xử lý của chương trình làm cho hệ thống bị treo dẫn đến từ chối dịch vụ (DoS) hay có khả năng bị các hacker lợi dụng chèn các chỉ thị trái phép nhằm thực thi các đoạn mã nguy hiểm từ xa. Có hai dạng lỗi tràn bộ đệm là **stack-based** và **heap-based**.

Cả hai thành phần stack và heap đều được sử dụng để lưu trữ các biến người dùng khi chạy chương trình. Khi một chương trình được nạp vào bộ nhớ được chia thành 6 giai đoạn tương ứng với sơ đồ phân đoạn trong bộ nhớ như hình minh họa bên dưới :

Hình 17.1 - Sơ đồ các phân đoạn trong bộ nhớ

Đầu tiên, các chỉ thị lệnh hay mã máy (phần tập tin thực thi nhị phân) sẽ được nạp qua phân đoạn text để thực thi các tác vụ của ứng dụng, vùng này được gán giá trị chỉ đọc có kích thước cố định tùy thuộc vào giá trị khởi tạo khi chương trình được nạp. Tiếp theo là phân doạn data chưa các biến toàn cục có giá trị khởi tạo ban đầu. Sau đó là vùng bss (below stack session) cũng dùng để lưu các biến toàn cục nhưng không có giá trị khởi tạo, kích thươc của vùng này và data cũng cố định khi chương trình được nạp. Và cuối cùng là vùng ENV, dùng để nạp các biến môi trường và đối số, cũng là giai đoạn sau cùng khi ứng dụng được nạp và thực thi.

Trong các phân đoạn trên thì phân doạn heap và stack là những nơi mà hacker sẽ tiến hành khai thác lỗi tràn bộ đệm, vùng heap dùng để cấp phát các biến động trong khi thực thi bởi các lời gọi hàm như malloc(). Heap phát triển từ vùng bộ nhớ có địa chỉ từ thấp đến cao theo nguyên tắt *FIFO (Firt in first out, biến nào nạp trước sẽ lấy ra sử dụng trước)*. Như hình dưới minh họa một nội dung của Heap :



Hình 17.2 – Một nội dung của heap

Khi một ứng dụng sao chép dữ liệu mà không kiểm tra kích thước có phù hợp với khả năng lưu trữ hay không thi hacker sẽ tận dụng để cung cấp những dữ liệu có kích thươc lớn làm tràn *heap* và ghi đè lên các biến động khác dẫn đến tình trạng **heap-based overflow.**

Còn vùng *stack* thì ngược lại dùng để lưu trữ các lời gọi hàm theo nguyên tắt *LIFO (Last in first out, lời gọi nào nạp vào sau sẽ được sử dụng trước)*. Những biến được lưu trữ trong các vùng này sẽ chờ cho đến khi nhận được lời gọi hàm để thực thi, và mội khi các biến này bị ghi đè bởi một chương trình nguy hiểm nào đó thì chương trình sẽ thực hiện chỉ thị này của hacker thông qua lời gọi hàm của mình, và tình huống bị khai thác lỗi như vậy gọi là *stack-based buffer overflow*.

Shell Code

Shellcode hay **paypload** là thuật ngữ dùng để chỉ những chương trình thường có kích thước khá nhỏ mà hacker sẽ chèn vào đúng các vị trí thực thi lệnh kế tiếp của con trỏ khi bị tràn bộ đệm. Với mục tiêu sẽ tiến hành các hành động mà hacker mong muốn như trong phần video minh họa một dạng tấn công lỗi tràn bộ đệm trên Windows XP tôi chọn *shell code* là nạp giao diện dòng lệnh trên máy tính bị tấn công, *shellcode* này có tên là reserver_shell, ngoài ra có nhiều loại shell code khác nhau đã được viết sẵn như chèn các dll mới lên máy tính bị tấn công, hay tạo tài khoản người dùng mới …

Các *shellcode* thường được viết bằng hợp ngữ và chèn trực tiếp vào các đoạn mã khai thác. Ví dụ vào ngày 26.3.2012 có một mã khai thác lỗi buffer overflow của UltraVNC 1.0.2 Client được công bố tại địa chỉ http://www.exploit-db.com/exploits/18666/ với shellcode là :

```
sploit = "\x52\x46\x42\x20\x30\x30\x33\x2e\x30\x31\x36\x0a" << [sploit.length].pack('N') << sploit
```

Hoặc các shellcode khác có dạng như :

```
"\x2d\x0b\xd8\x9a\xac\x15\xa1\x6e\x2f\x0b\xdc\xda\x90\x0b\x80\x0e"
"\x92\x03\xa0\x08\x94\x1a\x80\x0a\x9c\x03\xa0\x10\xec\x3b\xbf\xf0"
"\xdc\x23\xbf\xf8\xc0\x23\xbf\xfc\x82\x10\x20\x3b\xaa\x10\x3f\xff"
"\x91\xd5\x60\x01\x90\x1b\xc0\x0f\x82\x10\x20\x01\x91\xd5\x60\x01"
```

Hình 17.3 – Một đoạn shellcode

Các bước tiến hành khai thác buffer overflow

1. Tìm vị trí hay các điểm gấy ra lỗi tràn bộ đệm của ứng dụng.

2. Ghi các dữ liệu có kích thước lớn để vượt quá khả năng kiểm soát của chương trình.

3. Ghi đè lên địa chỉ trả về của các hàm.

4. Thay đổi chương trình thực thi bằng đoạn mã của hacker.

Như đoạn code bên dưới mô tả một tình huống bị lỗi *bof* của hàm *bof ()*, do kích thước *buffer* chi chứa tối đa 8 kí tự nhưng hàm *strcpy* sao chép đến 20 kí tự vào bộ nhớ vượt quá khả năng lưu trữ đã được khai báo trong bộ nhớ đệm.

```
/* This is a program to show a simple uncontrolled
overflow of the stack. It will overflow EIP with
0x41414141, which is AAAA in ASCII.  */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int bof(){
char buffer[8];
strcpy(buffer,"AAAAAAAAAAAAAAAAAAAA");
/*copy 20 bytes of A into the buffer*/
return 1; /*return, this will cause an access
violation due to stack corruption.*/ }
int main(int argc, char **argv){
bof(); /*call our function*/
/*print a short message, execution will never reach
this point because of the overflow*/
printf("Lets Go\n");
return 1; /*leaves the main function*/ }
```

Hình 17.4 – Một đoạn mã bị lỗi tại hàm bof()

Các bạn hãy tham khảo thêm một ví dụ về tràn bộ đệm viết bằng ngôn ngữ C là *overrun.c*

```
#include <stdio.h>
main() {
  char *name;
  char *dangerous_system_command;
  name = (char *) malloc(10);
  dangerous_system_command = (char *)
  malloc(128);
  printf("Address of name is %d\n", name);
  printf("Address of command is %d\n",
  dangerous_system_command);
  sprintf(dangerous_system_command, "echo
  %s", "Hello world!");
  printf("What's your name?");
  gets(name);
  system(dangerous_system_command);
}
```

Hình 17.5 – Một ví dụ khác về tràn bộ đệm

Trong phần đầu của đoạn mã sẽ khai báo hai biến kiểu chuỗi và gán bộ nhớ cho chúng. Tiếp theo biến name sẽ được cấp phát 10 byte trong bộ nhớ (có thể lưu tối đa 10 kí tự) còn biến dangerous_system_command được cấp phát đến 128 byte, như vậy hacker có thể chạy đè (overrun) lên vùng nhớ của biến name thông qua các giá trị nhập vào qua biến dangerous_system_command để thực thi các shellcode của mình (chúng ta sẽ thảo luận về chủ đề shell code ở phần tiếp theo)

Khi các bạn biên dịch đoạn mã o*verrun.c* trên linux sẽ cho kết quả như sau :

```
gcc overrun.c -o overrun
[XX]$ ./overrun
Address of name is 134518696\
Address of command is 134518712
What's your name?xmen
Hello world!\
[XX]$
```

Hình 17.6 – Kết quả biên dịch overrun.c

Như vậy, nếu như hacker nhập vào một biến có độ dài 16 kí tự thì sẽ bị tràn 6 kí tự cho phép thực thì các chỉ thị ngoài ý muốn như cat /etc/passwd trong hình 17.7 :

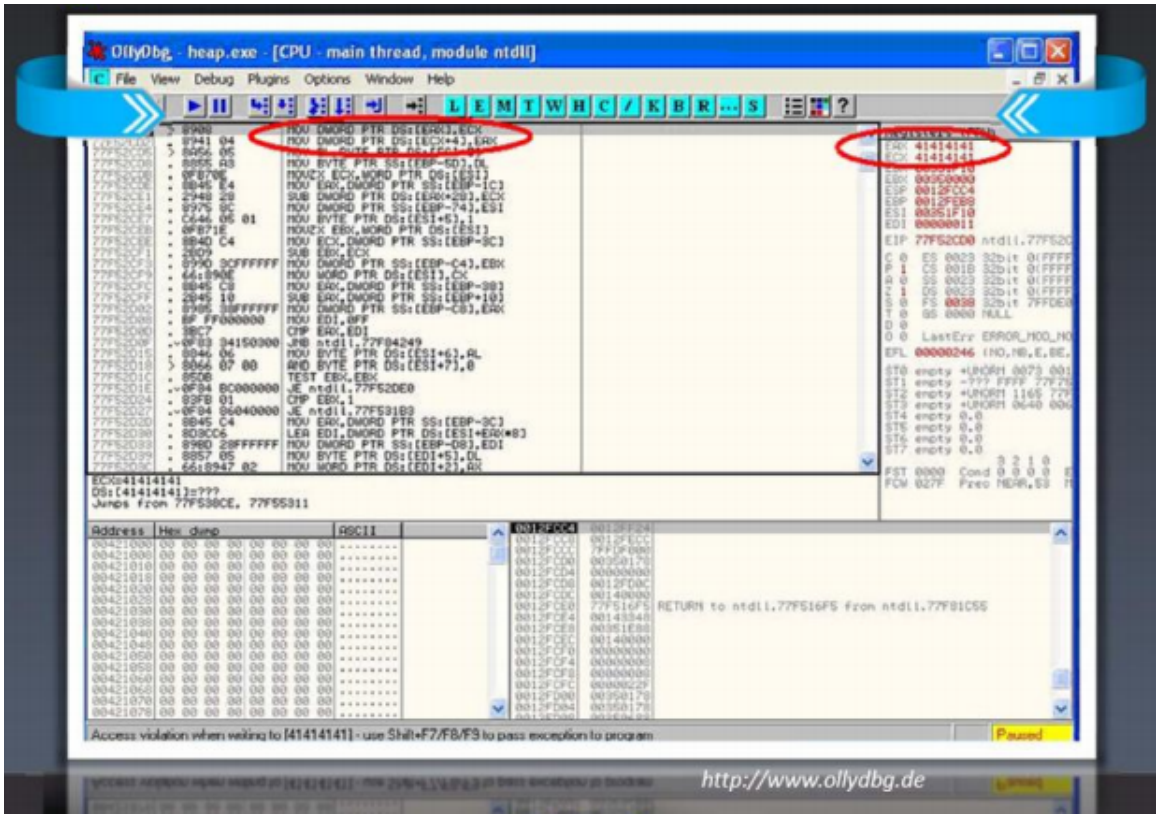
```
[XX]$ ./overrun
Address of name is 134518696
Address of command is 134518712
What's your
name?01234567891234567cat/etc/passwd

root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail
```

Hình 17.7 – Cung cấp biến gây lỗi tràn bộ đệm

Tìm Kiếm Lỗi Tràn Bộ Đệm Của Ứng Dụng

Để tìm kiếm các lỗi tràn bộ đệm hacker có thể dùng công cụ *BOU (Buffer Overfolw Utility)* kiểm tra xem các web site có bị lỗi *buffer overflow* hay không. Hay kiểm tra bằng công cụ *OllyDbg* như trong hình minh họa dưới sẽ xác định được các địa chỉ bị ghi đè.



Hình 17.8 - OllyDbg

Một quy trình kiểm tra ứng dụng mẫu sample.exe với OllDbg Debuger nhằm tìm xem có bị lỗi *BoF* :

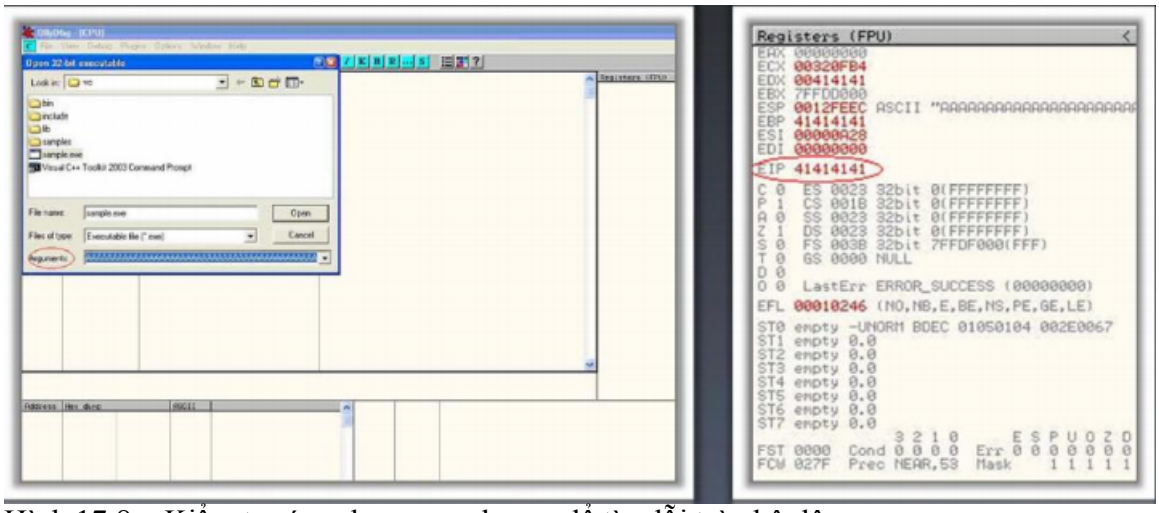
1 – Kiểm tra ứng dụng sample.exe (mở chương trình với OllDbg)

2 – Khởi động chương trình trong OllDbg

3 – Nạp vào một số lượng lớn các kí tự ví dụ AAAAAAAAAAA vào ứng dụng.

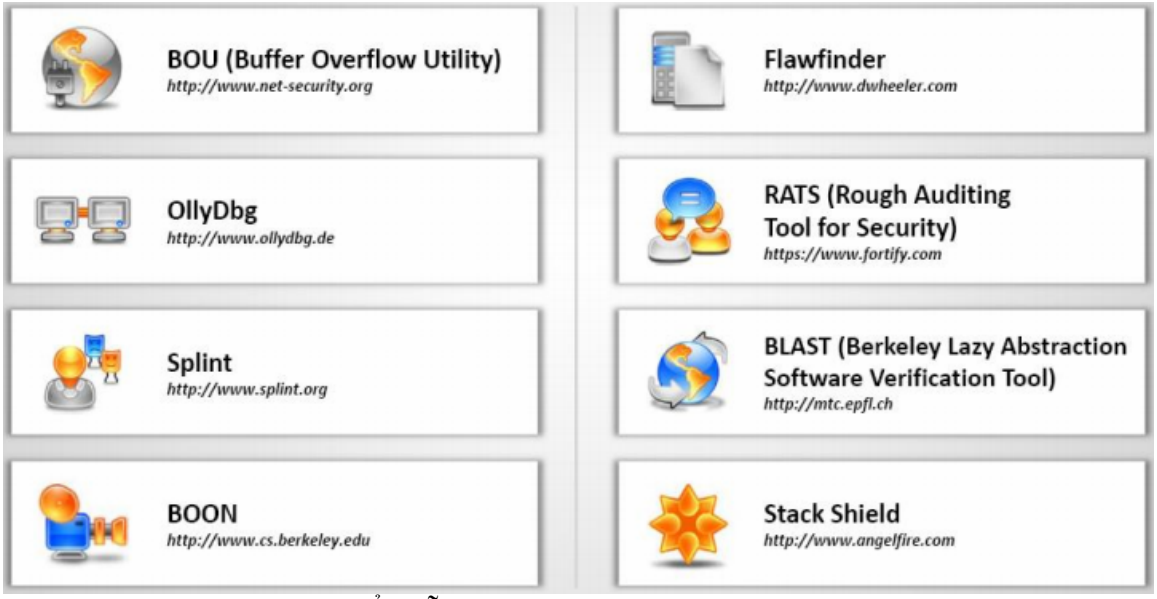
4 – Chạy chương trình với dữ liệu đầu vào đã nạp để kiểm tra các báo cáo của OllDBG

5 – EIP chưa giá trị 41414141 tượng trưng cho chuỗi AAAA trong hexa là các giá trị bị tràn.



Hình 17.8 – Kiểm tra ứng dụng sample.exe để tìm lỗi tràn bộ đệm

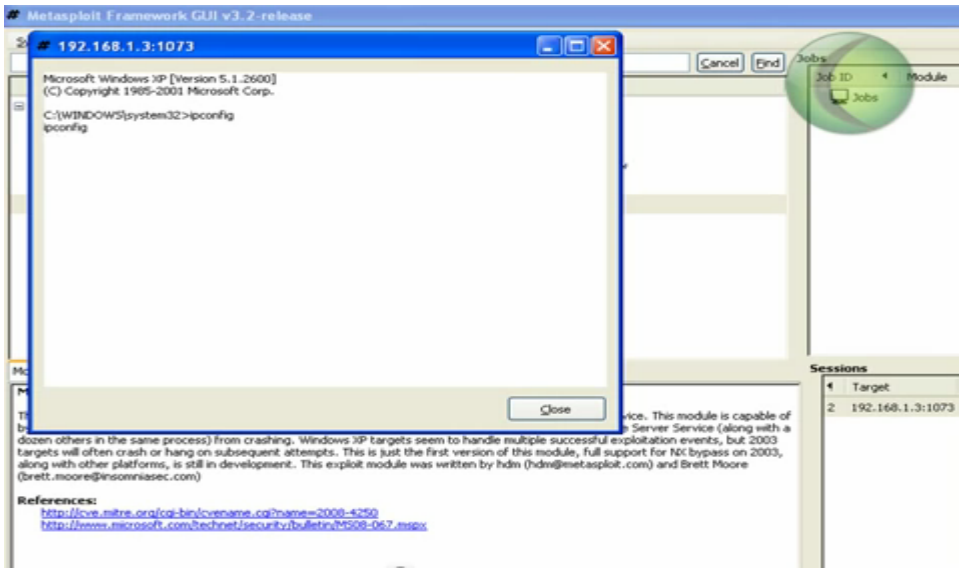
Ngoài ra, chúng ta còn có thể dùng IDA Pro để xác định các ứng dụng bị lỗi tràn bộ đệm, sau đây là danh sách các chương trình dùng để kiểm tra lỗi BoF :



Hình 17.9 – Các công cu6 kiểm lỗi Buffer Overflow

Minh Họa Khai Thác Lỗi Tràn Bộ Đệm

Khi hệ thống bị BoF các hacker có thể xác định các vị trí và giá trị gây lỗi để chèn shellcode vào chương trình, thông qua đó có thể chiếm quyền điều khiển từ xa. Những công cụ khai thác phổ biên hiện nay mà các hacker thường sử dụng là Metasploit Framework hay Core Impact, Backtrack. Trong video minh họa sau chúng ta sẽ sử dụng công cụ khai thác để chiếm quyền điều khiển một hệ thống dùng hệ điều hành Windows XP (http://www.youtube.com/watch?v=AKwyY2bfgv8)



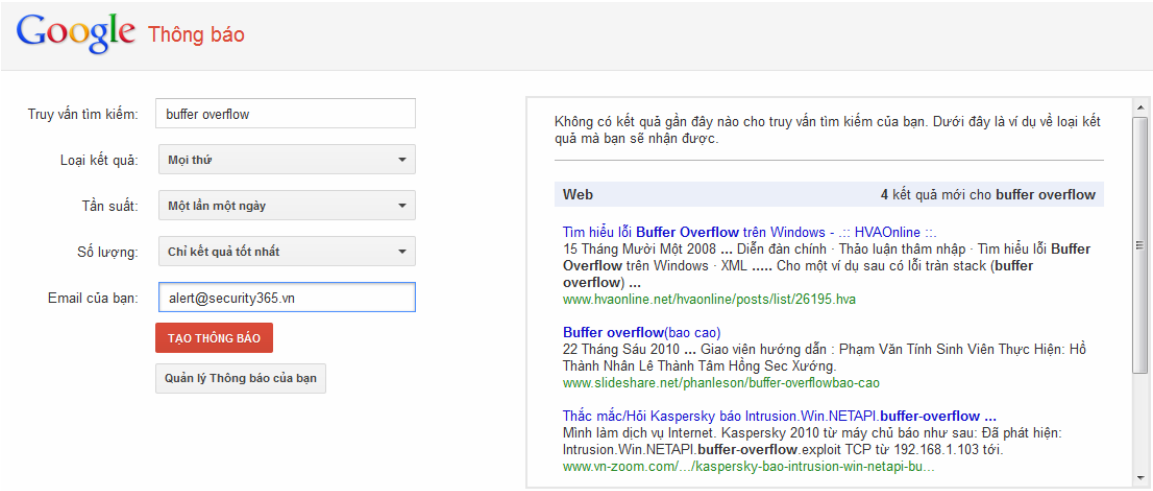
Hình 17.10 – Khai Thác Thành Công

Sau khi khai thác lỗi tràn bộ đệm của dịch vụ netapi trên Windows XP, hacker có thể chạylệnh trên máy nạn nhân qua payload reserve shell.

Tổng Kết

Trong chương này chúng ta đã tìm hiểu về quy trình nạp ứng dụng vào bộ nhớ và nguyên nhân dẫn đến tình trạng tràn bộ đệm, cần phân biệt trình tự xử lý của heap va stack. Các bạn lưu ý những công cụ mà hacker dùng để phát hiện lỗi cũng như các chương trình khai thác thông dụng.

Để không bị tấn công BoF thì các chương trình khi được viết cần tuân theo các quy tắt an toàn ngay từ lúc bắt đầu, kiểm tra lỗi cẩn thận với những chương trình debug như IDA Pro, OllDbg, đặc biệt quan tâm đến các hàm hay bị BoF như gets, strcpy …Bên cạnh đó khi biên dịch nên sử dụng các chương trình biên dịch an toàn như StackGuard nhằm ngăn chặn địa chỉ trả về bị ghi đè. Kiểm tra các biến đầu vào và hạn chế thực thi chương trình với quyền root (trên linux) hay quyền quản trị cao nhất trên Windows. Đặt các thông báo từ Google với từ khóa liên quan đến những chủ đề hay lỗi bảo mật cần quan tâm tại địa chỉ http://www.google.com/alerts như minh họa ..



Hình 17.11 – Đặt thông báo với Google Alert

Đối với hệ điều hành và các ứng dụng hoạt động trên máy chủ cần được thường xuyên kiểm tra bằng những chương trình như NESSUS, Retina, GFI nhằm phát hiện kịp thời các ứng dụng hay tiến trình gây lỗi. Trongvai trò quản trị hay một CEH thì chúng ta nên định kì cập nhập các bản vá, hotfix, theo dõi những thông tin mới nhất về lỗi bảo mật liên quan đến các ứng dụng hay hệ thống mà cơ quan, tổ chức đang dùng.