

# Symantec Antivirus Library Remote Heap Overflows Security Advisory

## **Date**

December 20, 2005

## **Vulnerability**

The Symantec Antivirus Library provides file format support for virus analysis. During decompression of RAR files Symantec is vulnerable to multiple heap overflows allowing attackers complete control of the system(s) being protected. These vulnerabilities can be exploited remotely without user interaction in default configurations through common protocols such as SMTP.

## **Impact**

Successful exploitation of Symantec protected systems allows attackers unauthorized control of data and related privileges. It also provides leverage for further network compromise. Symantec implementations are likely vulnerable in their default configuration. In default configurations users are likely vulnerable regardless of whether they choose to open or read the email.

## **Affected Products**

Due to the library's modular design and core functionality; it is likely this vulnerability affects a substantial portion of Symantec's gateway, server, & client antivirus-enabled product lines on most platforms. In fact, the scope of this vulnerability is likely similar to the one described in the following link and also includes more current versions.

<http://xforce.iss.net/xforce/alerts/id/187>

Further, this library is also licensed to a substantial number of vendors with products/services that are likely affected. A small sample of these vendors can be found in the following link.

<http://www.symantec.com/partners/index.html>

## **Recommendation**

Disable scanning of RAR compressed files until the vulnerable code is fixed.

## **Credit**

This vulnerability was discovered and researched by Alex Wheeler.

## **Contact**

security@remote.com

# Symantec Antivirus Library Remote Heap Overflows Security Advisory

## Description

The vulnerable code is responsible for decompression of RAR archive formats. Specifically, the vulnerability is the result of unchecked 16bit length fields in RAR sub-block header types. An attacker may craft a sub-block header to overwrite heap memory with user controlled file data to execute arbitrary code. Successful attack will yield system/root level privileges and is available through e-mail without user interaction.

The following vulnerable code is from Dec2Rar.dll v3.2.14.3. It's probable other versions are affected as well.

```
text:699DA96E      push     edi
text:699DA96F      call    getdecodedshort
text:699DA974      lea    edx, [ebp+2846h]
text:699DA97A      mov    ecx, ebx
text:699DA97C      push   edx
text:699DA97D      call    getdecodedbyte
text:699DA982      xor    eax, eax
text:699DA984      mov    ax, [edi]
text:699DA987      add    eax, 0FFFFFF00h ; switch 6 cases
text:699DA98C      cmp    eax, 5
text:699DA98F      ja     crcstuff      ; default
text:699DA995      jmp    ds:off_699DA004[eax*4] ; switch jump
text:699DA99C      case101: ; DATA XREF: .text:off_699DA004!o
text:699DA99C      lea    edi, [ebp+34F8h] ; case 0x101
text:699DA9A2      mov    ecx, 5
text:699DA9A7      rep movsd
text:699DA9A9      lea    esi, [ebp+350Ch]
text:699DA9AF      mov    ecx, ebx
text:699DA9B1      push   esi
text:699DA9B2      call    getdecodedshort
text:699DA9B7      lea    edi, [ebp+350Eh]
text:699DA9BD      mov    ecx, ebx
text:699DA9BF      push   edi
text:699DA9C0      call    getdecodedshort
text:699DA9C5      xor    eax, eax
text:699DA9C7      lea    ecx, [ebp+3510h] ; buf7890 off37E4
text:699DA9CD      mov    ax, [esi]
text:699DA9D0      push   eax ; len
text:699DA9D1      push   ecx ; buf
text:699DA9D2      mov    ecx, ebx
text:699DA9D4      call    getdecodedbytes__
text:699DA9D9      xor    edx, edx
text:699DA9DB      lea    eax, [ebp+3910h] ; buf7890 off37E4
text:699DA9E1      mov    dx, [edi]
text:699DA9E4      mov    ecx, ebx
text:699DA9E6      push   edx ; len
text:699DA9E7      push   eax ; buf
text:699DA9E8      call    getdecodedbytes__
text:699DA9ED      xor    ecx, ecx
text:699DA9EF      xor    edx, edx
text:699DA9F1      mov    cx, [esi]
text:699DA9F4      mov    byte ptr [ecx+ebp+3510h], 0
text:699DA9FC      mov    dx, [edi]
text:699DA9FF      mov    byte ptr [edx+ebp+3910h], 0
text:699DAA07      jmp    crcstuff      ; default
text:699DAA0C      ; -----
text:699DAA0C      case102: ; CODE XREF: _sub_699DA150_process_header_
text:699DAA0C      ; DATA XREF: .text:off_699DA004!o
text:699DAA0C      lea    edi, [ebp+3D10h] ; case 0x102
text:699DAA12      mov    ecx, 5
text:699DAA17      rep movsd
text:699DAA19      lea    eax, [ebp+3D24h]
text:699DAA1F      ;
```



# Symantec Antivirus Library Remote Heap Overflows Security Advisory

The above case can be reached through a sub-block type 1. It uses two unchecked shorts from the header as lengths for copy operations. The code below can be reached through sub-block type 5 and uses a third unchecked short as the length for a copy operation. In each instance the heap buffer size is 0x7890, however, the destination offsets differ and allow for an overflow a bit less than a short value.

```
text:699DAA5B      mov     ecx, ebx
text:699DAA5D      push   eax
text:699DAA5E      call  getdecodedbyte
text:699DAA63      lea   ecx, [ebp+3D45h]
text:699DAA69      push   ecx
text:699DAA6A      mov   ecx, ebx
text:699DAA6C      call  getdecodedbyte
text:699DAA71      lea   edx, [ebp+3D48h]
text:699DAA77      mov   ecx, ebx
text:699DAA79      push   edx
text:699DAA7A      call  getdecodedint
text:699DAA7F      jmp   crcstuff ; default
text:699DAA84      ;
-----
text:699DAA84      case105: ; CODE XREF: _sub_699DA150_process_header_type+845|j
text:699DAA84      ; DATA XREF: .text:off_699DA004|j
text:699DAA84      lea   edi, [ebp+3D4Ch]
text:699DAA8A      mov   ecx, 5
text:699DAA8F      rep  movsd
text:699DAA91      lea   eax, [ebp+3D60h]
text:699DAA97      mov   ecx, ebx
text:699DAA99      push   eax
text:699DAA9A      call  getdecodedint
text:699DAA9F      lea   ecx, [ebp+3D64h]
text:699DAAA5      push   ecx
text:699DAAA6      mov   ecx, ebx
text:699DAAA8      call  getdecodedbyte
text:699DAAA0      lea   edx, [ebp+3D65h]
text:699DAAB3      mov   ecx, ebx
text:699DAAB5      push   edx
text:699DAAB6      call  getdecodedbyte
text:699DAAB8      lea   eax, [ebp+3D68h]
text:699DAAC1      mov   ecx, ebx
text:699DAAC3      push   eax
text:699DAAC4      call  getdecodedint
text:699DAAC9      lea   esi, [ebp+3D6Ch]
text:699DAACF      mov   ecx, ebx
text:699DAAD1      push   esi
text:699DAAD2      call  getdecodedshort
text:699DAAD7      xor   ecx, ecx
text:699DAAD9      lea   edx, [ebp+3D6Eh] ; buf7890 off37E4
text:699DAADF      mov   cx, [esi]
text:699DAAE2      push   ecx ; len
text:699DAAE3      push   edx ; buf
text:699DAAE4      mov   ecx, ebx
text:699DAAE6      call  getdecodedbytes__
text:699DAAEB      xor   eax, eax
text:699DAAED      mov   ax, [esi]
text:699DAAF0      mov   byte ptr [eax+ebp+3D6Eh], 0
text:699DAAF8      jmp   short crcstuff ; default
text:699DAafa      ;
-----
text:699DAafa      casedefault: ; CODE XREF: _sub_699DA150_process_header_type+235|j
text:699DAafa      test  byte ptr [ebp+1BB9h], 80h ; default
text:699DAB01      iz   short crcstuff ; default
```

