THE GEOMETRY OF INNOCENT FLESH ON THE BONE: RETURN-INTO-LIBC WITHOUT FUNCTION CALLS

Published September 5, 2007

Hovav Shacham
hovav@cs.ucsd.edu

ACM Conf. on Computer and Communications Security (CCS 2007), pp. 552-561, 2007.

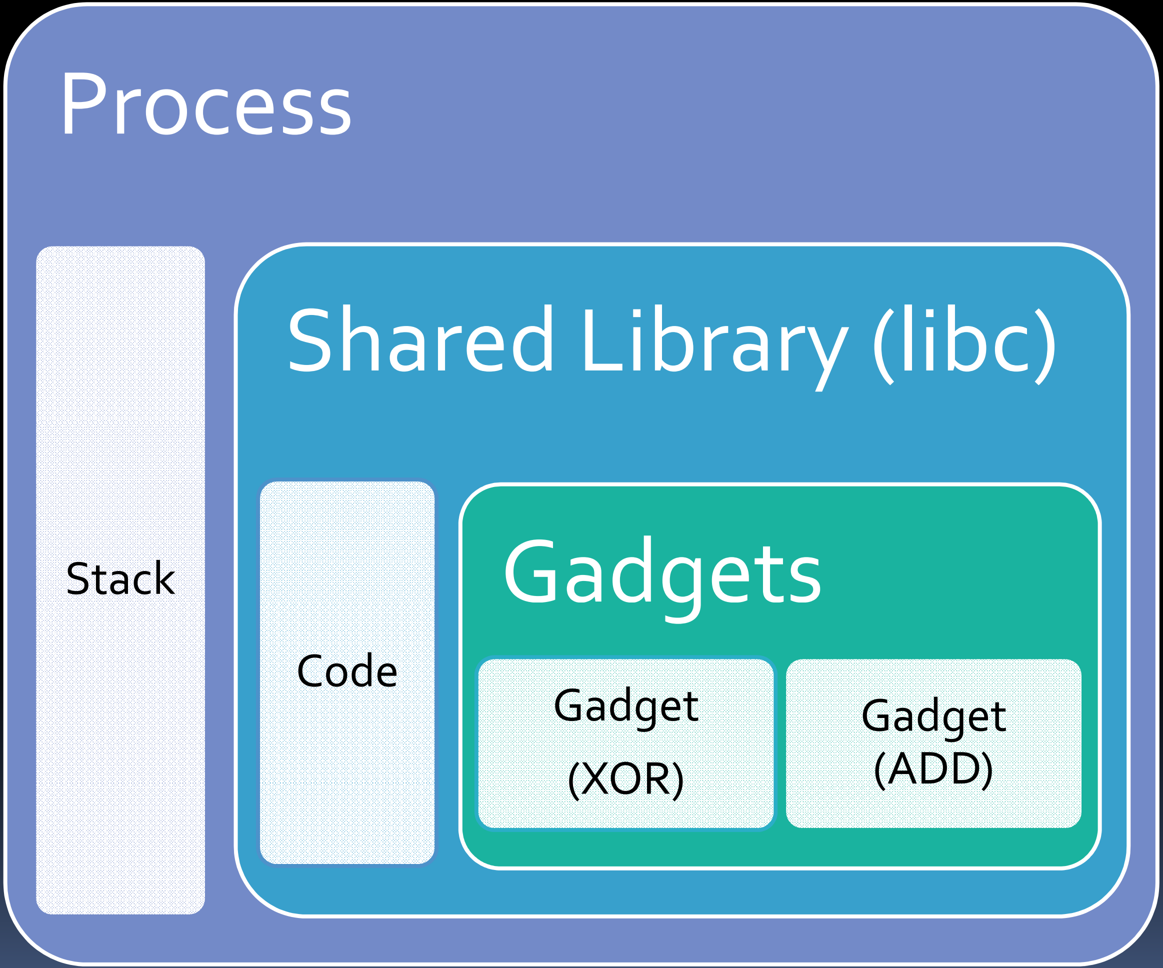
Presented by *James Restall*
August 25th 2008

Article Summary

The article shows how arbitrary code can be executed via a stack overflow exploit.

Specifically, by overwriting a function return address on the stack with a number of addresses inside the libc library. (return-into-libc)

It is shown how snippets of code from the libc library can build gadgets. The gadgets can then be used to execute any code sequence. (Turing complete)



Big Claims \ Limited Research

Critique

"In any sufficiently large body of x86 executable code there will exist sufficiently many useful code sequences [...] to cause the exploited program to undertake arbitrary computation."

- Claims in a large body of x86 code these gadgets can be made.
- Does not provide results for more than the linux c library.
- Windows? Mac?
  - Windows has dynamic library loading at unpredictable addresses.
- Other libraries?
- Various calling conventions? (stdcall, fastcall, thiscall…)

Presents Detailed Analysis

Compliment

- Comprehensive coverage of a Turing complete set of gadgets.
  - E.g. load constant, xor, add, shift, conditional jumps…
- Detailed examples from the libc library.
- Extensive explanation of each operation/gadget.
  - Diagrams of the stack setup for each.
- Proof of concept code provided for a libc attack.

Leaves no doubt that an attack would work using the libc code and modified return-into-libc attack.

Reveals Ineffective Security

Compliment

The article discloses a novel attack that renders current protections obsolete and ineffective.

Previous Protections

- Solar Designer’s StackPatch
- “WθX” – Linux PaX project
- Intel/AMD - Per-page execute disable bit.
- Removing functions from libraries.

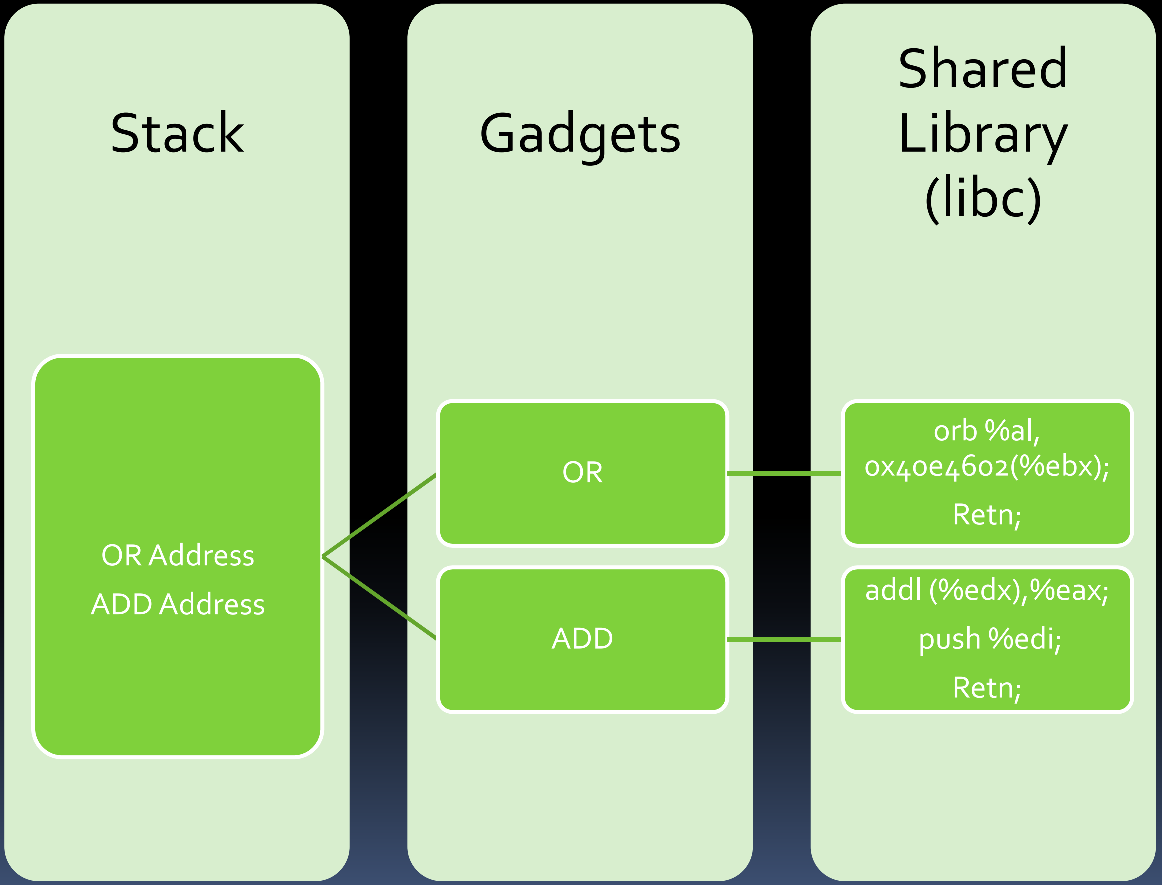
The new technique

New attack technique shows protections to be less useful than formerly thought.

Why so inadequate?

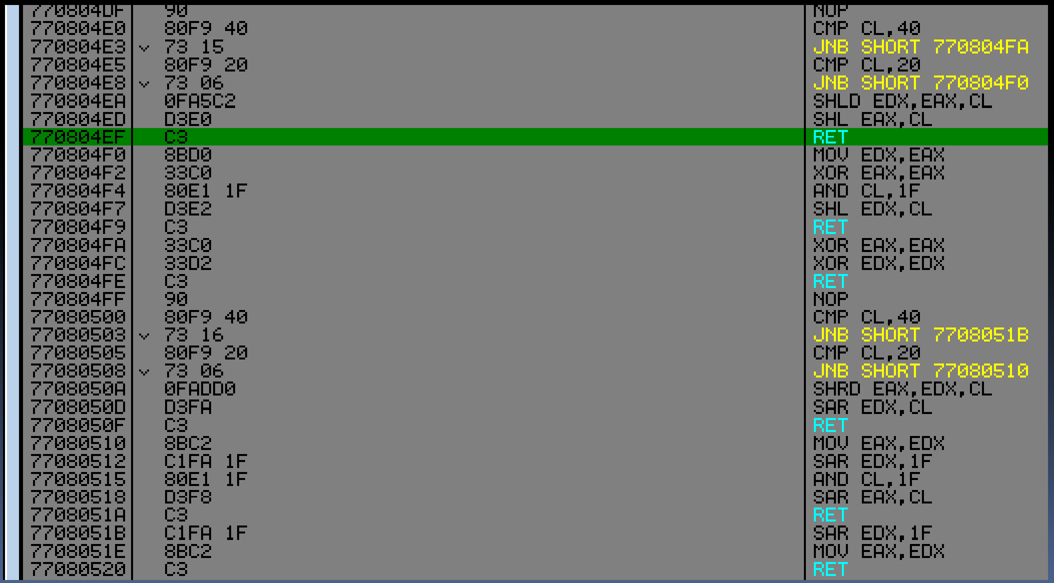
Attack doesn't execute on the stack – uses existing code. Therefore stack protection techniques irrelevant.

Attack runs arbitrary code – not existing functions. Therefore removing functions like system() from libraries doesn't help.


Stack
OR Address
ADD Address
Gadgets
OR
ADD
Shared Library (libc)
orb %al, 0x40e4602(%ebx); Retn;
addl (%edx),%eax; push %edi; Retn;

Question

Should code generators be changed to reduce the number of RET (return) instructions?



```
770804DF     90          NOP
770804E0     80F9 40     CMP CL,40
770804E3  v  73 15       JNB SHORT 770804FA
770804E5     80F9 20     CMP CL,20
770804E8  v  73 06       JNB SHORT 770804F0
770804EA     0FA5C2      SHLD EDX,EAX,CL
770804ED     D3E0        SHL EAX,CL
770804EF     C3          RET
770804F0     8BD0        MOV EDX,EAX
770804F2     33C0        XOR EAX,EAX
770804F4     80E1 1F     AND CL,1F
770804F7     D3E2        SHL EDX,CL
770804F9     C3          RET
770804FA     33C0        XOR EAX,EAX
770804FC     33D2        XOR EDX,EDX
770804FE     C3          RET
770804FF     90          NOP
77080500     80F9 40     CMP CL,40
77080503  v  73 16       JNB SHORT 7708051B
77080505     80F9 20     CMP CL,20
77080508  v  73 06       JNB SHORT 77080510
7708050A     0FADD0      SHRD EAX,EDX,CL
7708050D     D3FA        SAR EDX,CL
7708050F     C3          RET
77080510     8BC2        MOV EAX,EDX
77080512     C1FA 1F     SAR EDX,1F
77080515     80E1 1F     AND CL,1F
77080518     D3F8        SAR EAX,CL
7708051A     C3          RET
7708051B     C1FA 1F     SAR EDX,1F
7708051E     8BC2        MOV EAX,EDX
77080520     C3          RET
```