AWD

pwn

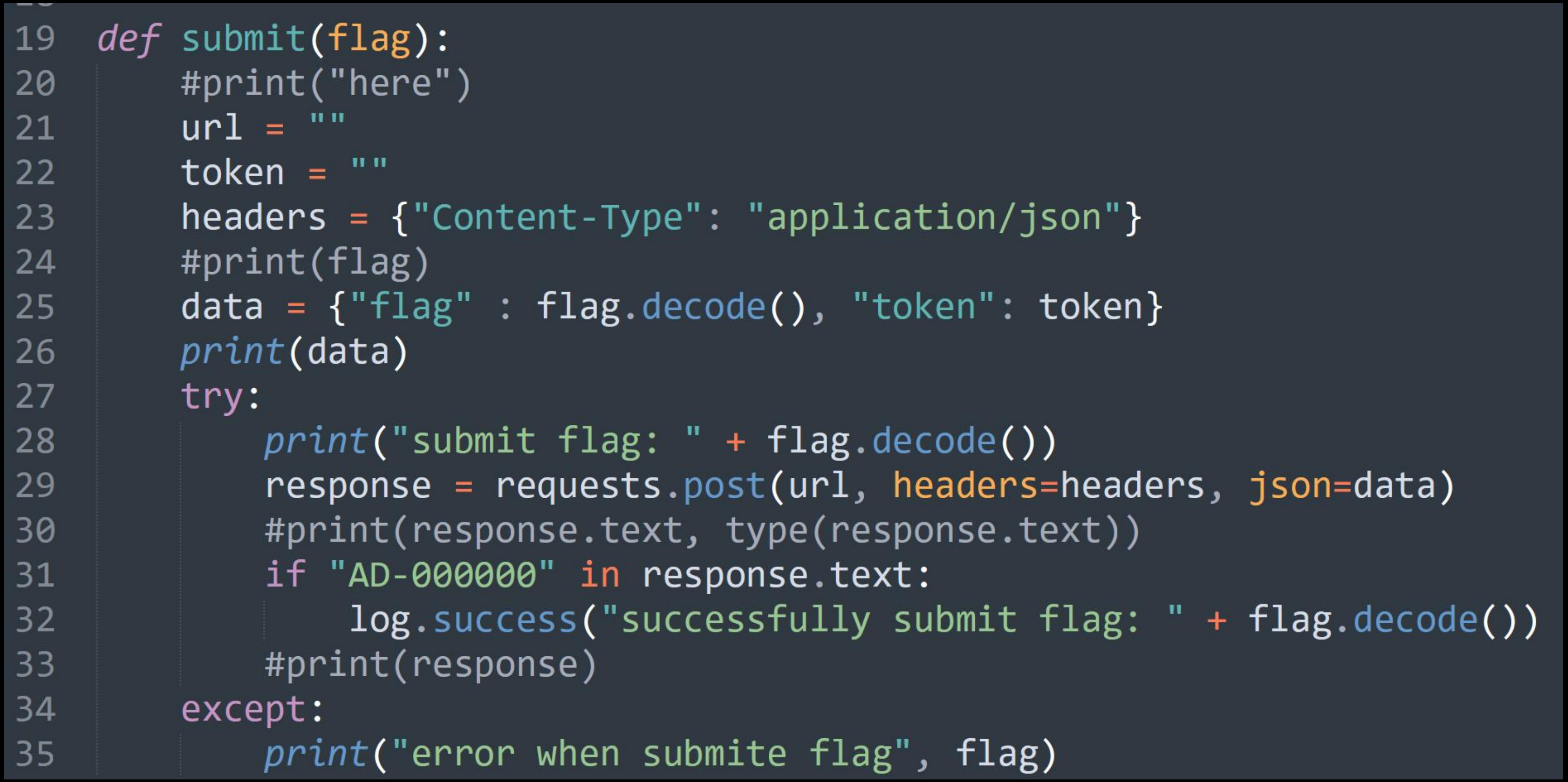
AWD

- 1. automatic exploit and defense
- 2. 1st demo - easyvm
- 3. 2nd demo - mqda
- 4. patch theory and practice

automatic exploit and defense

- automatic exploit:
- 1. finish the exploit
- 2. finish the submit flag function(remember to replace the token and url)
- 3. use sleep_ip_update.py to get flag and submit every XX seconds(remember to replace the ip and change the port)

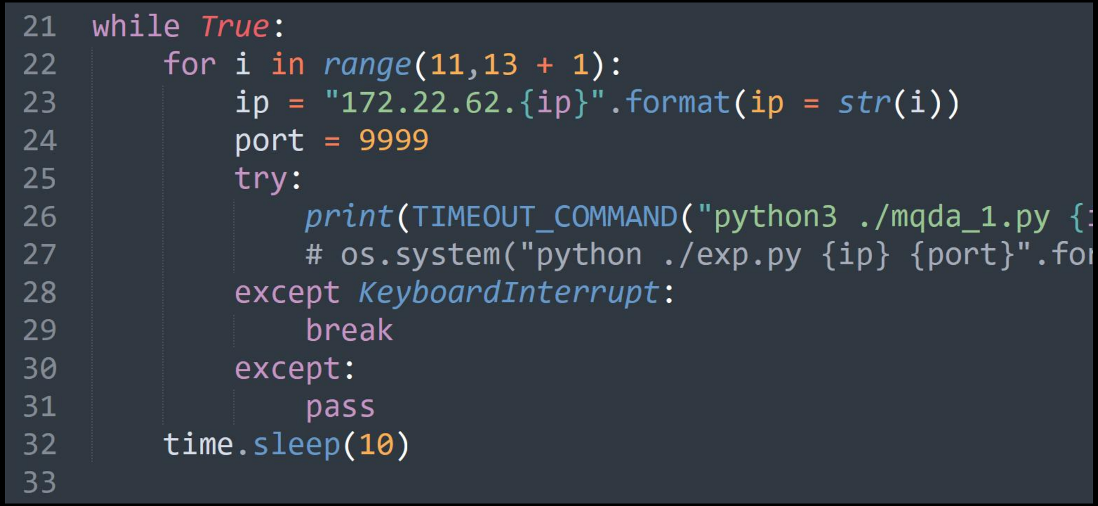
the submit flag function



```python
def submit(flag):
    #print("here")
    url = ""
    token = ""
    headers = {"Content-Type": "application/json"}
    #print(flag)
    data = {"flag" : flag.decode(), "token": token}
    print(data)
    try:
        print("submit flag: " + flag.decode())
        response = requests.post(url, headers=headers, json=data)
        #print(response.text, type(response.text))
        if "AD-000000" in response.text:
            log.success("successfully submit flag: " + flag.decode())
        #print(response)
    except:
        print("error when submite flag", flag)
```

sleep_ip_update.py

- Traverse ip from 172.22.62.11 to 172.22.62.13
- port is 9999
- every 10 seconds run this script

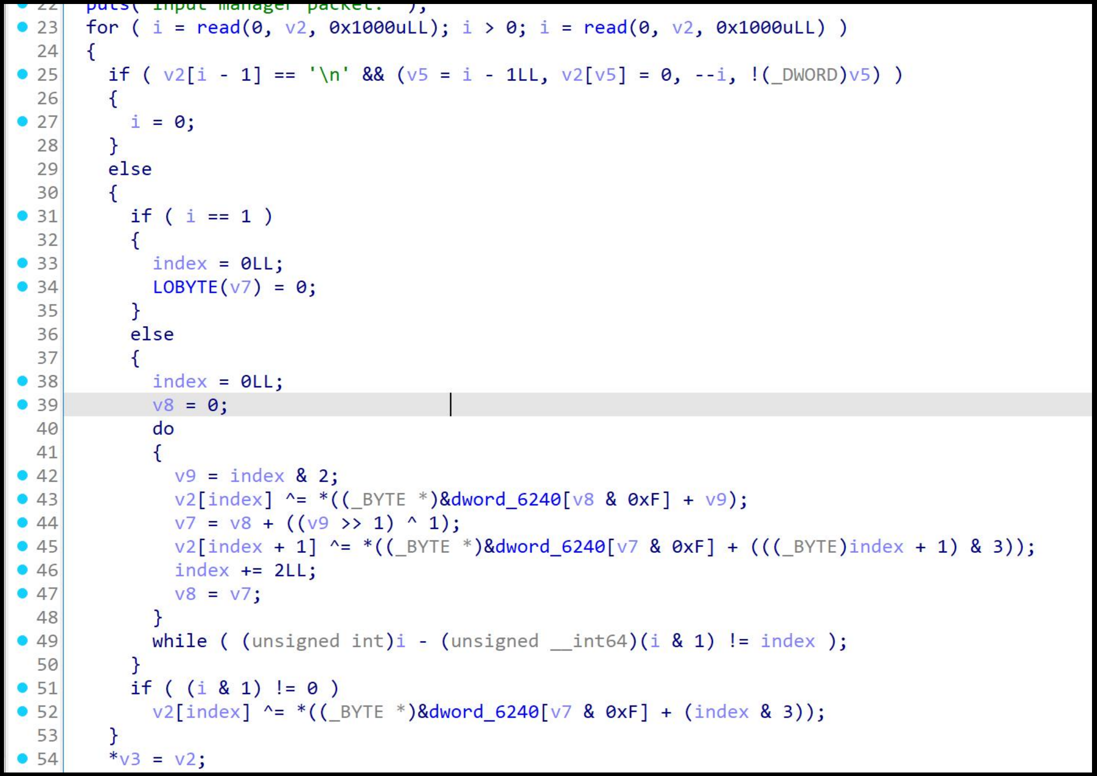


```python
while True:
    for i in range(11,13 + 1):
        ip = "172.22.62.{ip}".format(ip = str(i))
        port = 9999
        try:
            print(TIMEOUT_COMMAND("python3 ./mqda_1.py {
            # os.system("python ./exp.py {ip} {port}".fo
        except KeyboardInterrupt:
            break
        except:
            pass
    time.sleep(10)

```

1st demo - easyvm

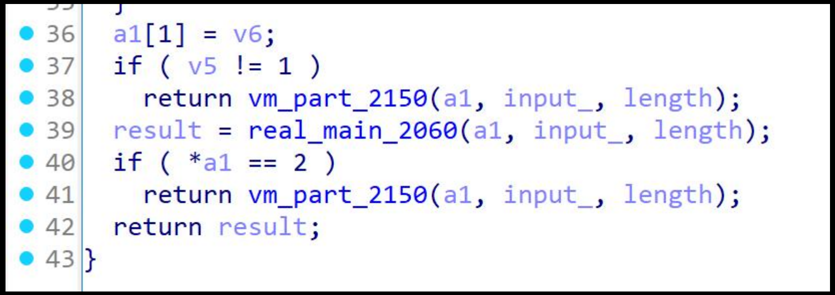
- first, use srand and rand to bypass the encrytion(just change the code from c to python)



```
puts("input manager packet:");
for ( i = read(0, v2, 0x1000uLL); i > 0; i = read(0, v2, 0x1000uLL) )
{
  if ( v2[i - 1] == '\n' && (v5 = i - 1LL, v2[v5] = 0, --i, !(_DWORD)v5) )
  {
    i = 0;
  }
  else
  {
    if ( i == 1 )
    {
      index = 0LL;
      LOBYTE(v7) = 0;
    }
    else
    {
      index = 0LL;
      v8 = 0;
      do
      {
        v9 = index & 2;
        v2[index] ^= *((_BYTE *)&dword_6240[v8 & 0xF] + v9);
        v7 = v8 + ((v9 >> 1) ^ 1);
        v2[index + 1] ^= *((_BYTE *)&dword_6240[v7 & 0xF] + (((_BYTE)index + 1) & 3));
        index += 2LL;
        v8 = v7;
      }
      while ( (unsigned int)i - (unsigned __int64)(i & 1) != index );
    }
    if ( (i & 1) != 0 )
      v2[index] ^= *((_BYTE *)&dword_6240[v7 & 0xF] + (index & 3));
  }
  *v3 = v2;
```

1st demo - easyvm

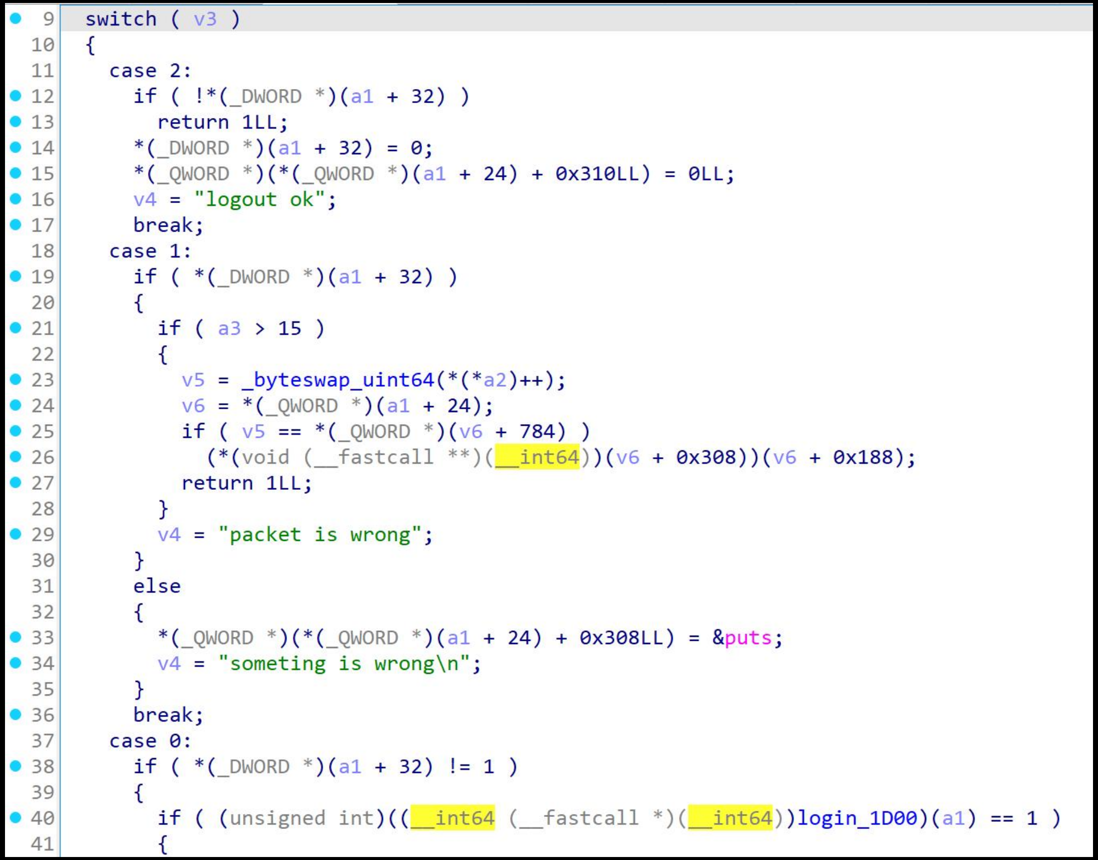
- two part
- 1. log in, log out, show
- 2. a vm



```
  a1[1] = v6;
  if ( v5 != 1 )
    return vm_part_2150(a1, input_, length);
  result = real_main_2060(a1, input_, length);
  if ( *a1 == 2 )
    return vm_part_2150(a1, input_, length);
  return result;
}
```

1st demo - easyvm

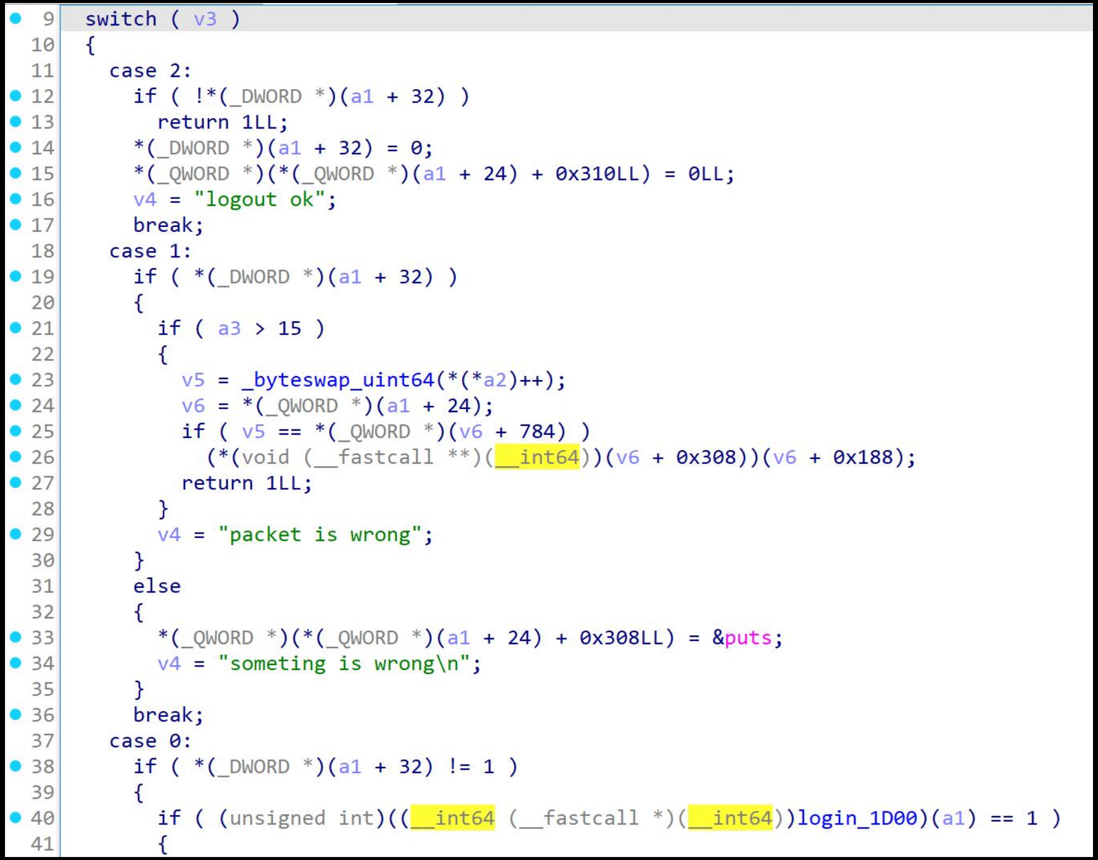
- overflow vulnerability in login_1D00
- show the puts pointer's address and overwrite it to get shell



```
switch ( v3 )
{
  case 2:
    if ( !*(_DWORD *)(a1 + 32) )
      return 1LL;
    *(_DWORD *)(a1 + 32) = 0;
    *(_QWORD *)(*(_QWORD *)(a1 + 24) + 0x310LL) = 0LL;
    v4 = "logout ok";
    break;
  case 1:
    if ( *(_DWORD *)(a1 + 32) )
    {
      if ( a3 > 15 )
      {
        v5 = _byteswap_uint64(*(*a2)++);
        v6 = *(_QWORD *)(a1 + 24);
        if ( v5 == *(_QWORD *)(v6 + 784) )
          (*(void (__fastcall **)(__int64))(v6 + 0x308))(v6 + 0x188);
        return 1LL;
      }
      v4 = "packet is wrong";
    }
    else
    {
      *(_QWORD *)(*(_QWORD *)(a1 + 24) + 0x308LL) = &puts;
      v4 = "someting is wrong\n";
    }
    break;
  case 0:
    if ( *(_DWORD *)(a1 + 32) != 1 )
    {
      if ( (unsigned int)((__int64 (__fastcall *)(__int64))login_1D00)(a1) == 1 )
      {
```

1st demo - easyvm

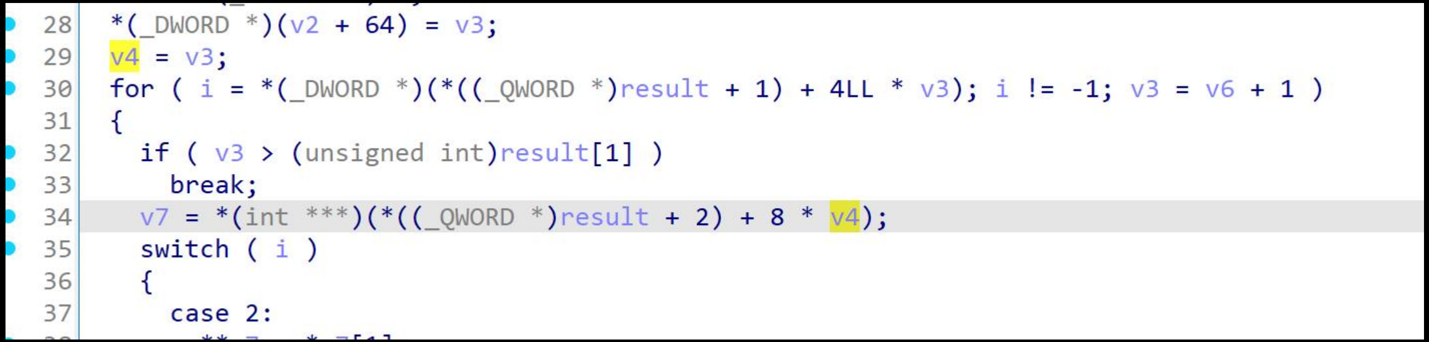
- overflow vulnerability in login_1D00
- show the puts pointer's address and overwrite it to get shell



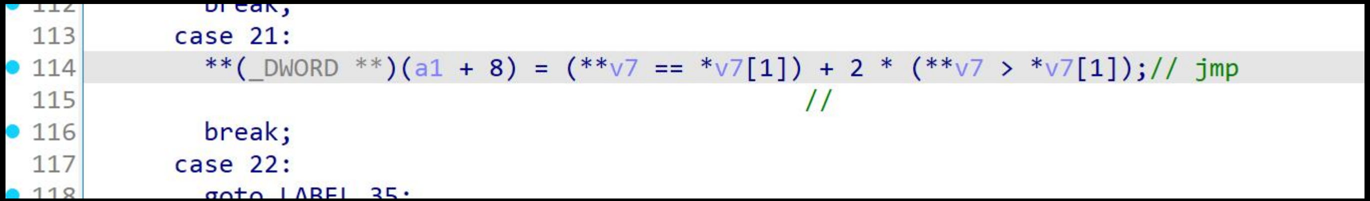
```
switch ( v3 )
{
  case 2:
    if ( !*(_DWORD *)(a1 + 32) )
      return 1LL;
    *(_DWORD *)(a1 + 32) = 0;
    *(_QWORD *)(*(_QWORD *)(a1 + 24) + 0x310LL) = 0LL;
    v4 = "logout ok";
    break;
  case 1:
    if ( *(_DWORD *)(a1 + 32) )
    {
      if ( a3 > 15 )
      {
        v5 = _byteswap_uint64(*(*a2)++);
        v6 = *(_QWORD *)(a1 + 24);
        if ( v5 == *(_QWORD *)(v6 + 784) )
          (*(void (__fastcall **)(__int64))(v6 + 0x308))(v6 + 0x188);
        return 1LL;
      }
      v4 = "packet is wrong";
    }
    else
    {
      *(_QWORD *)(*(_QWORD *)(a1 + 24) + 0x308LL) = &puts;
      v4 = "someting is wrong\n";
    }
    break;
  case 0:
    if ( *(_DWORD *)(a1 + 32) != 1 )
    {
      if ( (unsigned int)((__int64 (__fastcall *)(__int64))login_1D00)(a1) == 1 )
      {
```

1st demo - easyvm

- 1. Inadequate checks on negative index(v4(mov instruction), case 21(jmp instruction))
- 2. case arbitrary read and write(mov); arbitrary jump(case 0x666 branch to launch a shell)



```
*(_DWORD *)(v2 + 64) = v3;
v4 = v3;
for ( i = *(_DWORD *)(*((_QWORD *)result + 1) + 4LL * v3); i != -1; v3 = v6 + 1 )
{
  if ( v3 > (unsigned int)result[1] )
    break;
  v7 = *(int ***)(*((_QWORD *)result + 2) + 8 * v4);
  switch ( i )
  {
    case 2:
```

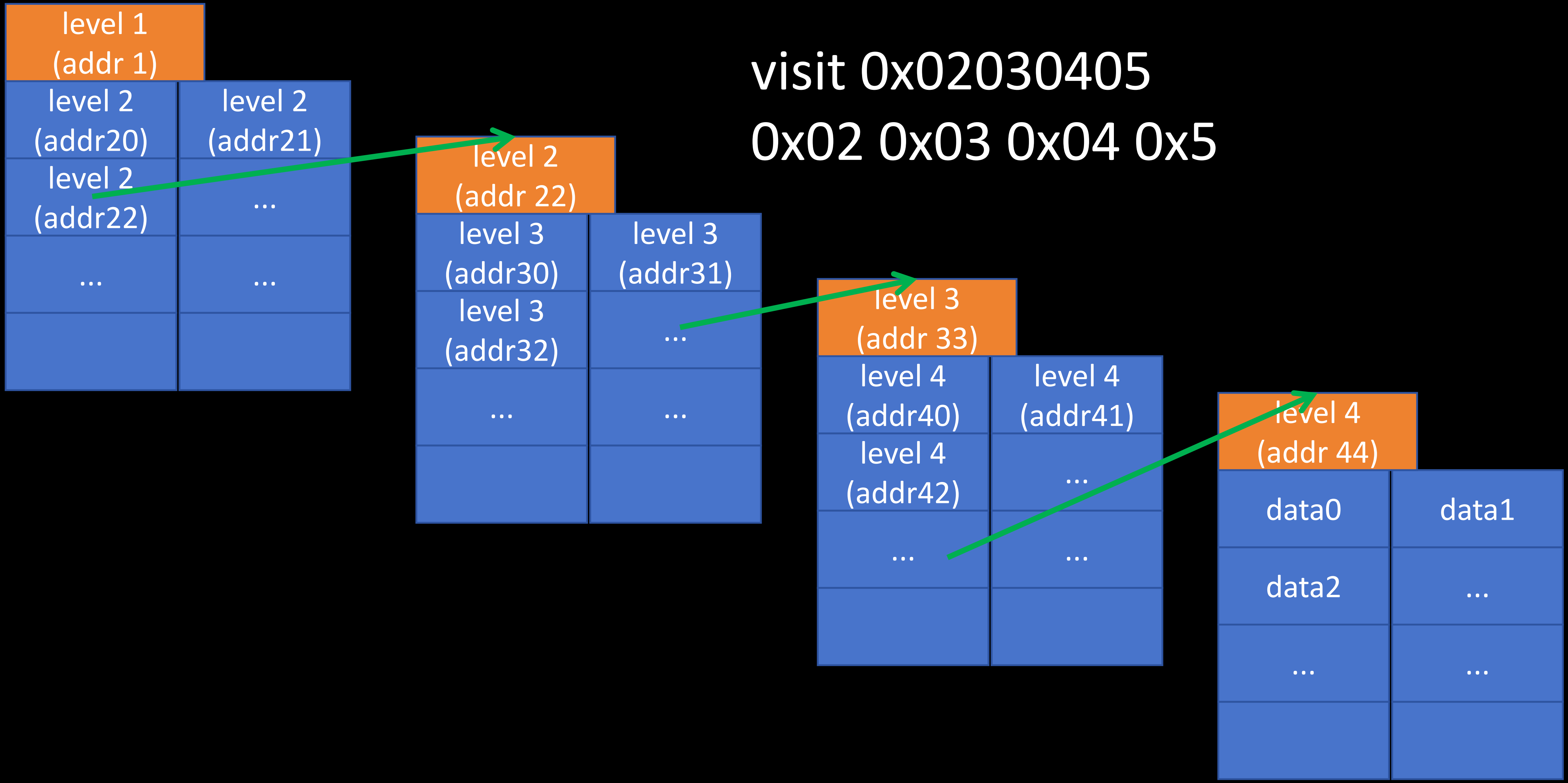


```
    case 21:
      **(_DWORD **)(a1 + 8) = (**v7 == *v7[1]) + 2 * (**v7 > *v7[1]);// jmp
                                                                        //
      break;
    case 22:
```

2nd demo - mqda

- have a easy access backdoor
- 4-level Page Tables
- a vm
- use malloc, every heap has esidual information

4-level Page Tables - visit 0x02030405 as example



visit 0x02030405

0x02 0x03 0x04 0x5

vulnerability

- esidual information from heap
- use esidual information to leak libc and heap
- use esidual information to fake a page table to realize arbitrary write and read

patch theory and practice

- 1. change data
- 2. add segements
- 3. compress instruction
- 4. add logicS