Programmatically NOP the Current Selection in Ghidra – nullteilerfrei

By born

Published: 2023-10-01 · Archived: 2026-04-05 23:02:59 UTC

malware-analysis, technology 2020-08-23

The Zlob malware contains lots of bogus API calls to hinder analysis. This blag post describes how to use a script in Ghidra to automate the process of patching out those calls with `NOP` instructions.

The malware employs some basic obfuscation techniques, one of which is performing lots of API calls without side effects. This makes both the disassembly as well as the decompile view hard to read. The following excerpts where take from a random sample belonging to this family (SHA256 hash `0b38ca277bbb042d43bd1f17c4e424e167020883526eb2527ba929b2f0990a8f`):

```
 ff d5 | CALL EBP=>KERNEL32.DLL::GetCurrentThreadId
ff d5 | CALL EBP=>KERNEL32.DLL::GetCurrentThreadId
ff d6 | CALL ESI=>KERNEL32.DLL::GetLastError
ff d7 | CALL EDI=>KERNEL32.DLL::GetConsoleCP
ff d6 | CALL ESI=>KERNEL32.DLL::GetLastError
ff d6 | CALL ESI=>KERNEL32.DLL::GetLastError
ff d6 | CALL ESI=>KERNEL32.DLL::GetLastError
ff d7 | CALL EDI=>KERNEL32.DLL::GetConsoleCP
ff d6 | CALL ESI=>KERNEL32.DLL::GetLastError
ff d7 | CALL EDI=>KERNEL32.DLL::GetConsoleCP
ff d6 | CALL ESI=>KERNEL32.DLL::GetLastError
```

And the corresponding decompile view:

```
 GetCurrentThreadId();
GetCurrentThreadId();
GetLastError();
GetConsoleCP();
GetLastError();
GetLastError();
GetLastError();
GetConsoleCP();
GetLastError();
GetConsoleCP();
```

```
GetLastError();
[...]
```

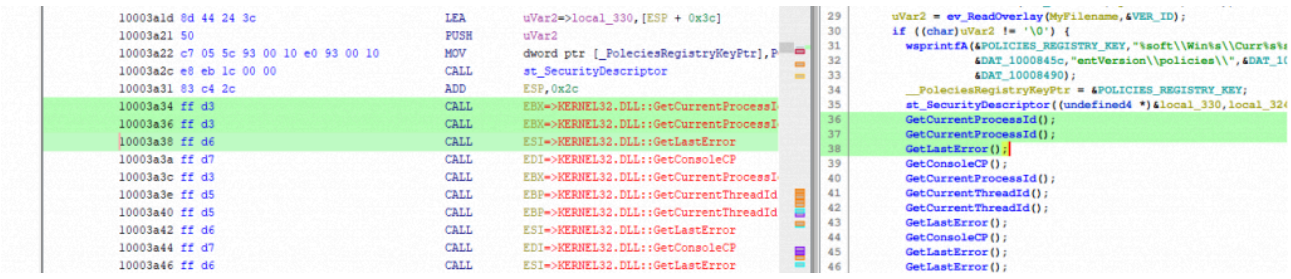
You can "just" hit Ctrlf+Shift+G after selecting every call (and the following byte) to replace it with the `NOP` instruction (`0x90`) but this is a more than tedious process (it would be even more tedious to right click into the Listing view and select "Patch Instruction"). Let's write a handy script to automate this:

```
 public void run() throws Exception {
    if (currentSelection != null) {
        AddressRangeIterator addressRanges = currentSelection.getAddressRanges(true);
        for (AddressRange addressRange : addressRanges) {
            nopOut(addressRange.getMinAddress(), addressRange.getLength());
        }
    }
}

private void nopOut(Address addressStart, long length) throws CancelledException, MemoryAccessException {
    clearListing(addressStart, addressStart.add(length));
    for (int i = 0; i < length; i++) {
        Address address = addressStart.add(i);
        setByte(address, (byte) 0x90);
    }

    disassemble(addressStart);
}
```

If you assign this script to the keyboard shortcut Ctrl-Alt-Shift-N for example, you can just select regions of disassembly and hit that combination to replace everything with `NOP` .

A tiny plot-twist

Since selecting calls in the decompile view will also highlight disassembly, one might think that you can as easily select region of the decompile view, execute the same script and come to the same result. Sadly this is not the case: If you select a line in the decompile view, the corresponding selection in the disassembly will always only have length 1.



https://blag.nullteilerfrei.de/wp-content/uploads/2020/08/disassembly-selection-1024x212.png 1024w, https://blag.nullteilerfrei.de/wp-content/uploads/2020/08/disassembly-selection-300x62.png 300w,

https://blag.nullteilerfrei.de/wp-content/uploads/2020/08/disassembly-selection-768x159.png 768w, https://blag.nullteilerfrei.de/wp-content/uploads/2020/08/disassembly-selection.png 1313w" sizes="(max-width: 1024px) 100vw, 1024px">

While one might think that in the screenshot above, the selection in the disassembly has a lenght of 6 and ranges from 0x10003a34 to 0x10003a3a (exclusively) it is in fact three selections, each of length 1 starting at `0x10003a34`, `0x10003a36` and, `0x10003a38`. Executing the above script will hence result in the byte sequence `90 d3 90 d3 90 d3` which cannot even be disassembled.

So let's not pass in `addressRange.getLength()` into the `nopOut` function but instead extend the length such that the last instruction is always included completely:

```
 /**
 * Searchers backwards for the last assembly instruction and returns the length
 * of the address range, potentially extended to fully include this last
 * instruction.
 */
private long assemblyAlignedLength(AddressRange addressRange) {
    long length = addressRange.getLength();
    for (int i = 1; i <= MAX_ASSEMBLY_INSTRUCTION_LENGTH; i++) {
        Instruction instruction = getInstructionAt(addressRange.getMinAddress().add(length - i));
        if (instruction != null) {
            return length + (instruction.getLength() - i);
        }
    }

    return length;
}
```

[You can find the complete `nts.java` script on github!]

Edit (2023-10-01): F2 is not correct, Ctrl-Shift-G it is.

Source: https://blag.nullteilerfrei.de/2020/08/23/programmatically-nop-the-current-selection-in-ghidra/