8 On Error Resume Next

by Jeffball

Don't you just long for the halcyon days of Visual Basic 6 (VB6)? Between starting arrays at `1` and only needing signed data types, Visual Basic was just about as good as it gets. Well, I think it's about time we brought back one of my favorite features: `On Error Resume Next`. For those born too late to enjoy the glory of VB6, `On Error Resume Next` allowed those courageous VB6 ninjas who dare wield its mightiness to continue executing at the next instruction after an exception. While this may remove the pesky requirement to handle exceptions, it often caused unexpected behavior.

When code crashes in Linux, the kernel sends the `SIGSEGV` signal to the faulting program, commonly known as a segfault. Like most signals, this one too can be caught and handled. However, if we don't properly clean up whatever caused the segfault, we'll return from that segfault just to cause another segfault. In this case, we simply increment the saved `RIP` register, and now we can safely return. The third argument that is passed to the signal handler is a pointer to the user-level context struct that holds the saved context from the exception.

```
void segfault_sigaction(int signal, siginfo_t *si, void * ptr) {
  ((ucontext_t *)ptr)->uc_mcontext.gregs[REG_RIP]++;
}
```

Now just a little code to register this signal handler, and we're good to go. In addition to `SIGSEGV`, we'd better register `SIGILL` and `SIGBUS`. `SIGILL` is raised for illegal instructions, of which we'll have many since our On Error Resume Next handler may restart a multi-byte instruction one byte in. `SIGBUS` is used for other types of memory errors (invalid address alignment, non-existent physical address, or some object specific hardware errors, etc) so it's best to register it as well.

```
  struct sigaction sa;
  memset(&sa, 0, sizeof(sigaction));
  sigemptyset(&sa.sa_mask);
  sa.sa_sigaction = segfault_sigaction;
  sa.sa_flags   = SA_SIGINFO;

  sigaction(SIGSEGV, &sa, NULL);
  sigaction(SIGILL, &sa, NULL);
  sigaction(SIGBUS, &sa, NULL);
```

In order to help out the users of buggy software, I've included this code as a shared library that registers these handlers upon loading. If your developers are too busy to deal with handling errors or fixing bugs, then this project may be for you. To use this code, simply load the library at runtime with the `LD_PRELOAD` environment variable, such as the following:

```
$ LD_PRELOAD=./liboern.so ./login
```

Be wary though, this may lead to some unexpected behavior. The attached example shell server illustrates this, but can you figure out why it happens?[51]

```
$ nc localhost 5555
Please enter the password:
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
↪     AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Password correct, starting access shell...
```

[51] `unzip pocorgtfo08.pdf onerror.zip #Beware of spoilers!`