17:01 I thought I turned it on, but I didn't.

Neighbors, please join me in reading this eighteenth release of the International Journal of Proof of Concept or Get the Fuck Out, a friendly little collection of articles for ladies and gentlemen of distinguished ability and taste in the field of reverse engineering and the study of weird machines. This release is a gift to our fine neighbors in Leipzig and Washington, D.C.

If you are missing the first seventeen issues, we suggest asking a neighbor who picked up a copy of the first in Vegas, the second in São Paulo, the third in Hamburg, the fourth in Heidelberg, the fifth in Montréal, the sixth in Las Vegas, the seventh from his parents' inkjet printer during the Thanksgiving holiday, the eighth in Heidelberg, the ninth in Montréal, the tenth in Novi Sad or Stockholm, the eleventh in Washington D.C., the twelfth in Heidelberg, the thirteenth in Montréal, the fourteenth in São Paulo, San Diego, or Budapest, the fifteenth in Canberra, Heidelberg, or Miami, the sixteenth release in Montréal, New York, or Las Vegas, or the seventeenth release in São Paulo or Budapest.

After our paper release, and only when quality control has been passed, we will make an electronic release named `pocorgtfo17.pdf`. It is a valid PDF document and a ZIP file filled with fancy papers and source code. It is also a valid program for the Apollo Guidance Computer, which will run in the VirtualAGC emulator.

As you'll recall from PoC||GTFO 3:11, AES in CBC mode allows you to flip bits of the initialization vector to flip bits of the first cleartext block. On page 5, Albert Spruyt and Niek Timmers share some handy tricks for using a similar property: by flipping bits of one block's ciphertext you can also flip blocks of the subsequent ciphertext block after decryption. In this manner, they can sacrifice half of the blocks by flipping their bits to control the other half, loading shellcode into the cleartext of an encrypted ARM image for which they have no key.

Our own Pastor Laphroaig has a sermon for you on page 9, concerning the good ol' days of juvenile science fiction, when chemistry sets were dangerous and Dr. Watson trusty pistol was always at hand.

Software defined radios and radios built from custom hardware can receive damned near anything these days, but some of the most clever radio hacking involves firmware patches to existing, commodity radios. On page 13, Damien Cauquil shows us how to write custom firmware for the nRF51 chip in the BBC Micro:Bit to sniff an ongoing Bluetooth Low Energy connection, without previously knowing the hop interval, increment, or even the channel map.

Speaking of PHY layer tricks, what does a clever neighbor do when he hasn't got a hardware PHY? For Ethernet, Andrew Zonenberg simply bitbangs it from an old Spartan-6 FPGA and the right resistors. Page 21.

When assembling hardware, sometimes it can be ambiguous whether a chip is inserted one way, or rotated one hundred and eighty degrees from that way. On page 32, Joe Grand shares with us a DIP-8 design that selectively re-adjusts itself to having the chip rotated. Build your PCB by the ferric chloride method with a 0.1" DIP socket for proper nostalgia.

Back in the good ol' days, folks would share hooking techniques over a pint of good ale. Now that pints have as few as eight ounces, and some jerk ranting about Bitcoin ruins all our conversations, it's nice to read that Shawn Webb has been playing with methods for hooking functions in FreeBSD processes through unprivileged `ptrace()` debugging. Page 34.

Page 42 features a gumshoe detective novella, one in which Soldier of Fortran hangs out his neon sign and teams up with Bigendian Smalls to create the niftiest EBCDIC login screen for his z/OS mainframe.

Leandro Pereira has some clever tricks on page 56 for injecting additional code into preexisting ELF files to enable defensive features through `seccomp-bpf`.

On page 60, the last page, we pass around the collection plate. Our church has no interest in bitcoins or wooden nickels, but we'd love your donation of a reverse engineering story. Please send one our way.

WRIST MAC™

Wouldn't it be great—

to have your telephone book, your appointment schedule and your To Do list available instantly, 24 hours a day?
How about daily reminders? Multiple alarm clocks? Price list information? Project details? Client phone numbers?

The WristMac™ is a high quality digital watch that talks to your Macintosh!

The **WristMac™** downloads up to 80 screen pages of your most important information in less than 30 seconds. Your data can be quickly imported from your existing Macintosh files, including Apple's Hypercard stacks, Focal Point II, QuickDEX, Dynodex, Address Book Plus, Smart Alarms, plain text files, and many others. Once the information is in the watch, it can be edited and transferred back to the Mac, using the optional bi-directional adapter!

The **WristMac™** is a complete system, including watch, cable and software. It adapts to the way you work: use it as a stand-alone system for keeping track of your personal information, or use the easy import ability to pick up your existing data.

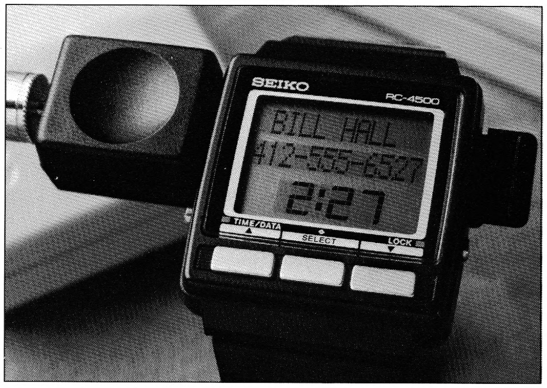
Now you CAN take it with you!

Watch Features:

- State of the art digital watch with day, date, hours, minutes, seconds
- Additional screen shows two 12-character lines. Timed memos sound alarm and display a 12 character message
- Phone memo shows 12 character name plus phone number
- Free-form text displays 80 screens of 24 characters, divided among up to 12 different headings
- Included cable connects to Mac Plus, Classic, Portable, SE, SE/30, II, IIx, IIcx, IIci, SI and LC

Software Features:

- New version 2.0 Wristmac software
- Includes Apple's Hypercard 2.0 software free!
- Can be used as a stand-alone system
- Stores and recalls multiple "master lists"
- Extensive on-line help facility
- Imports from Apple's Address and Appointment stacks
- Imports from Focal Point II (seven different stacks)
- Imports from Activision's City to City and Business Class
- Imports from Portfolio System's Dynodex
- Imports from Power Up Software's Address Book Plus
- Imports from Jam Software's Smart Alarms
- Imports from Casady & Greene's QuickDex
- Imports from ACIUS' 4th Dimension
- Imports from any tab-delimited text file
- Exports to Survivor Software's MacMoney accounting system
- Exports to tab-delimited text files



Suggested Retail Prices:

Standard WristMac™ (Black, Red, Green, Yellow, Gray)	$145
Executive WristMac™ Black	195
Pocket WristMac™	195
Executive WristMac™ Gold or Silver	245
Bidirectional Adapter	75
Watch-to-Watch Transfer Adapter	25
WristMac™ Software and Cable Kit only	75

To Order:
For fast phone service, call **813-882-8635** or fax **813-884-5941** from 9:00am to 5:30 pm EST, Monday through Friday.
Or order by mail, from **Microseeds Publishing, Inc.**
5901 Benjamin Center Drive - Suite 103 • Tampa, FL 33634.
Payment by check, money order, Visa or MasterCard.

17:02 Constructing AES-CBC Shellcode

by Albert Spruyt and Niek Timmers

Howdy folks!

Imagine, if you will, that you have managed to bypass the authenticity measures (i.e., secure boot) of a secure system that loads and executes an binary image from external flash. We do not judge, it does not matter if you accomplished this using a fancy attack like fault injection[1] or the authenticity measures were lacking entirely.[2] What's important here is that you have gained the ability to provide the system with an arbitrary image that will be happily executed. But, wait! The image will be decrypted right? Any secure system with some self respect will provide confidentiality to the image stored in external flash. This means that the image you provided to the target is typically decrypted using a strong cryptographic algorithm, like AES, using a cipher mode that makes sense, like Cipher-Block-Chaining (CBC), with a key that is not known to you!

Works of exquisite beauty have been made with the CBC-mode of encryption. Starting with humble tricks, such as bit flipping attacks, we go to heights of dizzying beauty with the padding-oracle-attack. However, the characteristics of CBC-mode provide more opportunities. Today, we'll apply its bit-flipping characteristics to construct an image that decrypts into executable code! Pretty nifty!

Cipher-Block-Chaining (CBC) mode

The primary purpose of the CBC-mode is preventing a limitation of the Electronic Code Book (ECB) mode of encryption. Long story short, the CBC-mode of encryption ensures that plain-text blocks that are the same do not result in duplicate cipher-text blocks when encrypted. Below is an ASCII art depiction of AES decryption in CBC-mode. We denote a cipher text block as $\texttt{CT}_i$ and a plain text block as $\texttt{PT}_i$.

```
                CT-1            CT-2
                  |_______        |_______ . . .
                  |       |       |
              _________   |   _________
             |         |  |  |         |
IV ---       |   AES   |  |  |   AES   |
        |    |_________|  |  |_________|
        |         |       |       |
        |______XOR        |______XOR
                  |               |
                PT-1            PT-2
```

An important aspect of CBC-mode is that the decryption of $\texttt{CT}_2$ depends, besides the AES decryption, on the value of $\texttt{CT}_1$. Magically, without knowing the decryption key, flipping 1 or more bits in $\texttt{CT}_1$ will flip 1 or more bits in $\texttt{PT}_2$.

Let's see how that works, where $\wedge 1$ denotes flipping a bit at an arbitrary position.

$$\texttt{CT}_1 \wedge 1 + \texttt{CT}_2$$

Which get decrypted into:

$$\texttt{TRASH} + \texttt{PT}_2 \wedge 1$$

[1] Bypassing Secure Boot using Fault Injection, Niek Timmers and Albert Spruyt, Black Hat Europe 2016

[2] Arm9LoaderHax — Deeper Inside, Jason Dellaluce