CORE SECURITY

Vulnerability Hunting in Access Controls

Bobby Kuzma, CISSP
Systems Engineer

October 21, 2016

About this talk

This talk is:

- A moderately technical discussion of abusing an inexpensive commodity access control device

This talk is not:

- An introduction
- Approved by Marketing

CORE SECURITY

Hi! I'm Bobby.

I show people how to use things. Like Pentesting Software.

I get to pentest things.

I break stuff and call it research.

And I love my job.

CORE SECURITY

A couple of things to start with

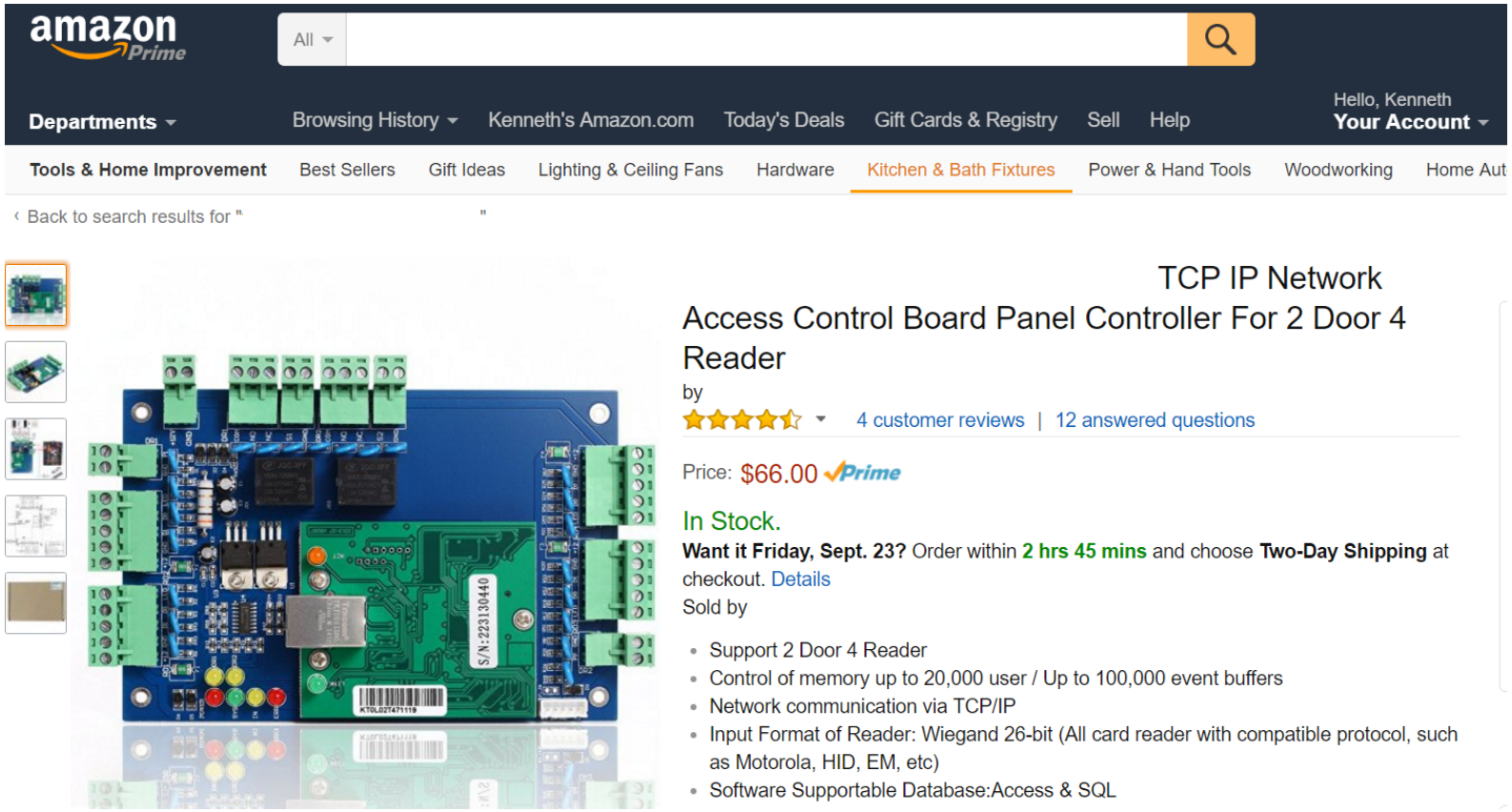
A word on Disclosure

- Core Labs operates under a Coordinated Disclosure Policy
- The Vendor has been notified of the vulnerabilities we'll be talking about
- We will NOT be naming the Vendor publicly until they release fixes. If you suspect that you have these in your environment, contact me after the talk.

CORE SECURITY

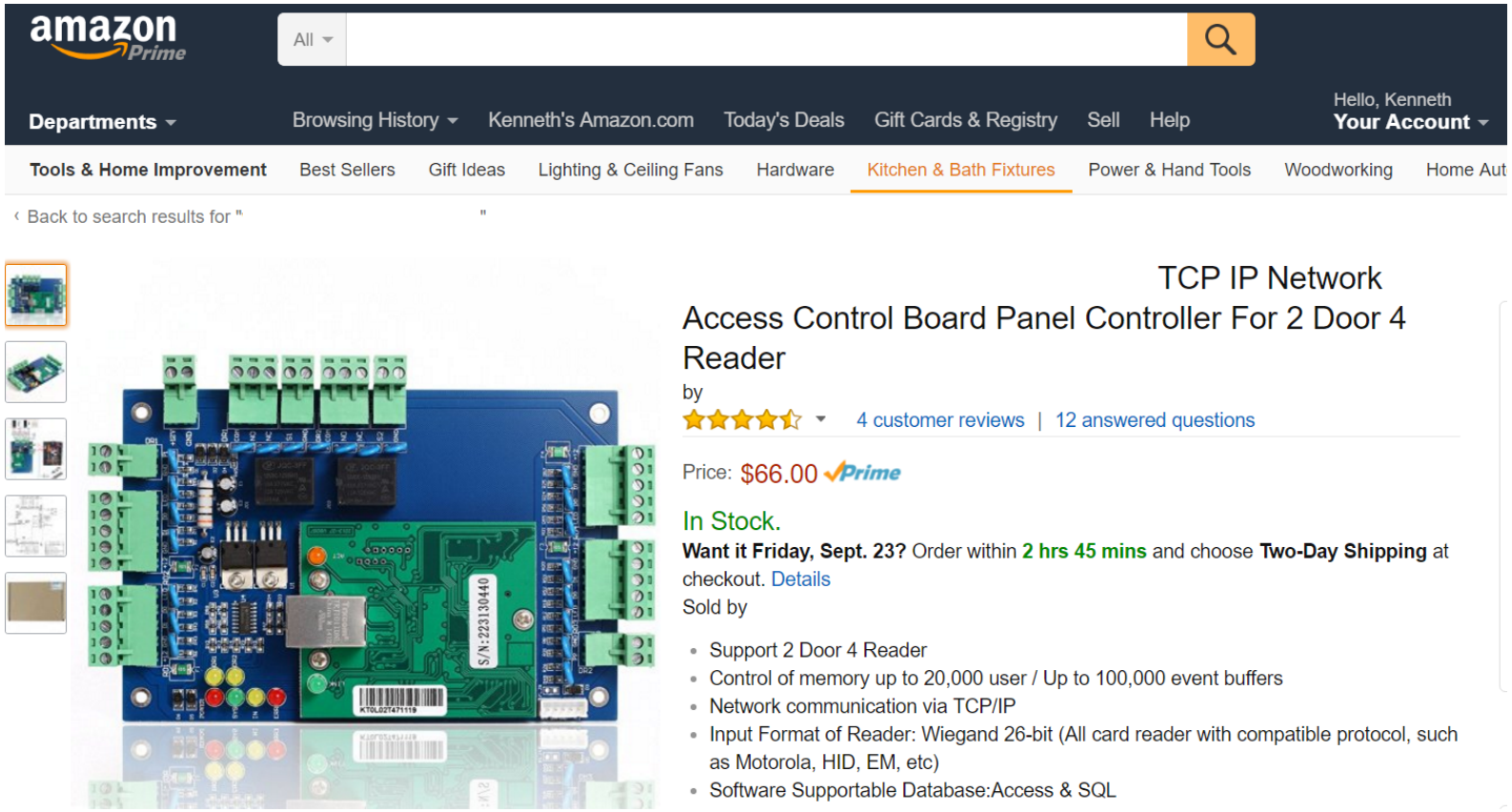
The Target

Commonly sold on Amazon, eBay, Alibaba, etc

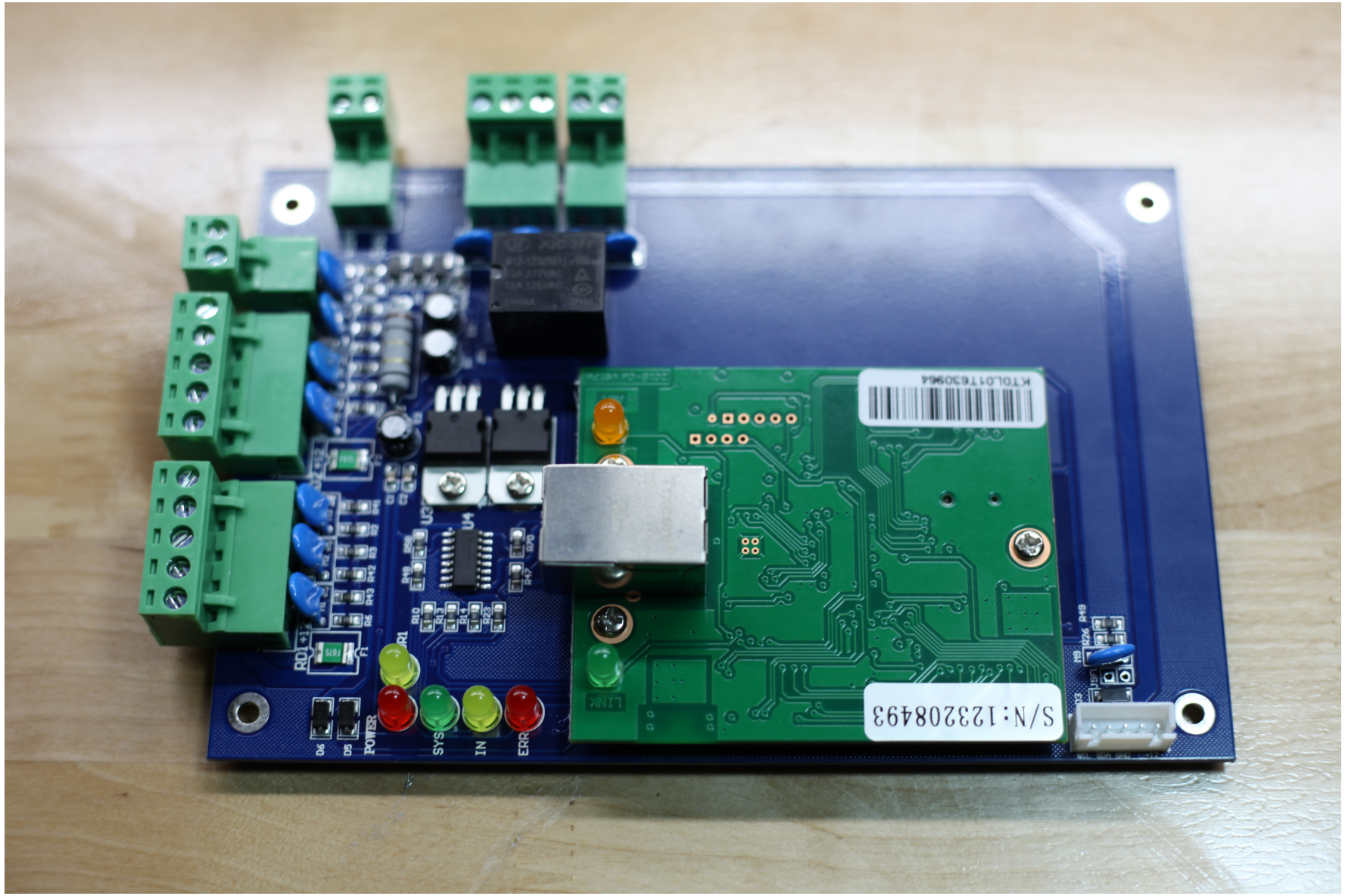


The Target

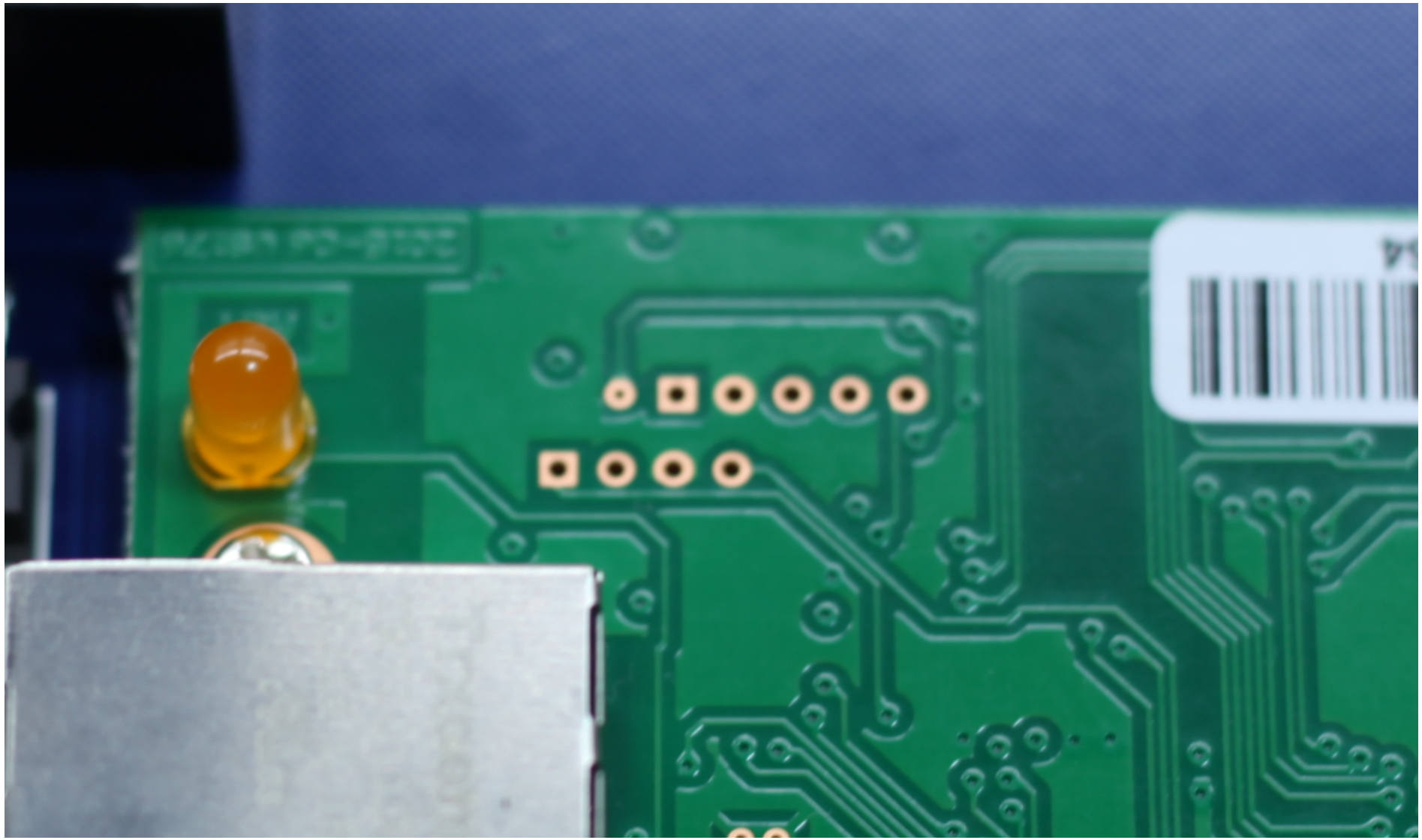
Available in 1, 2, or 4 door flavors



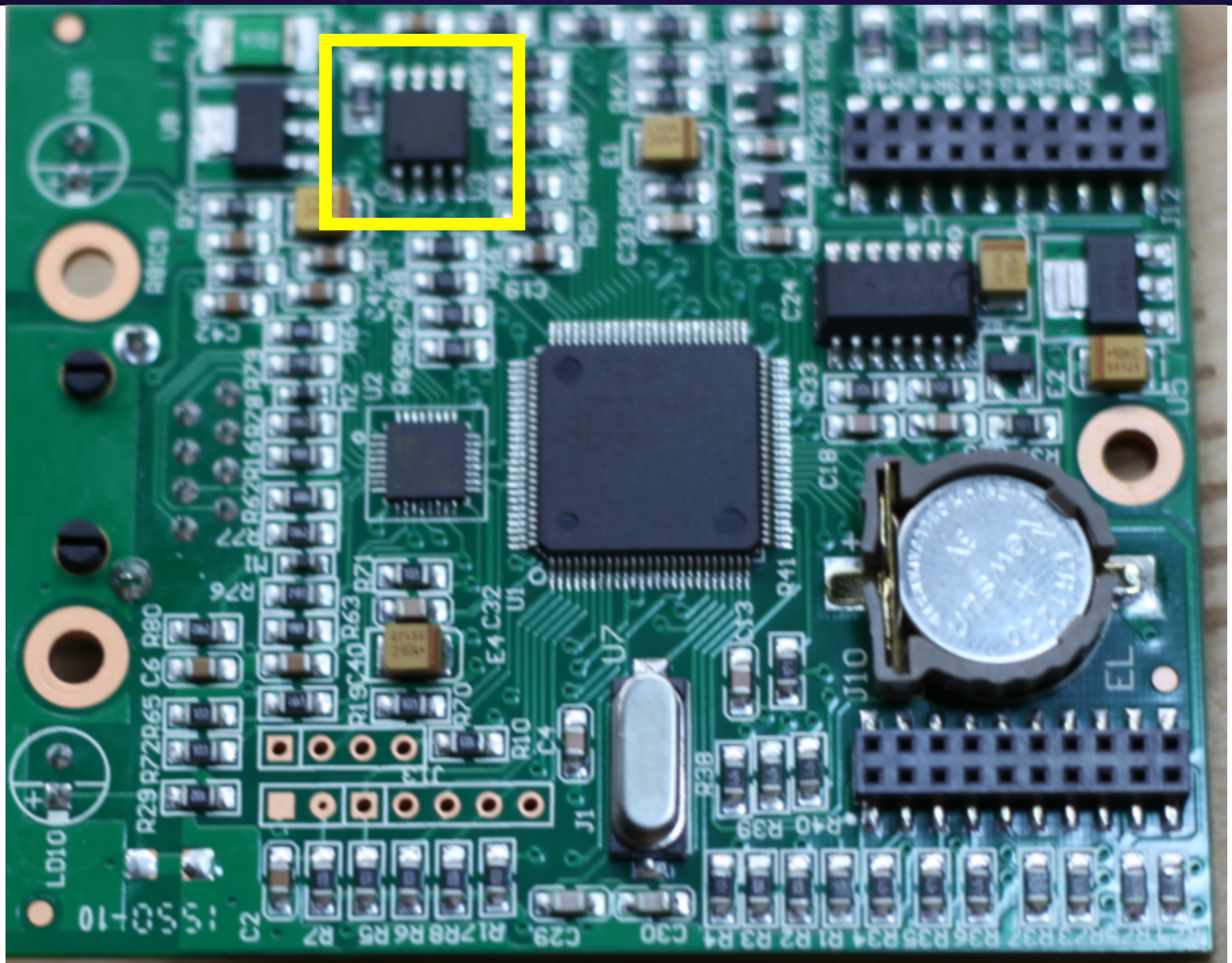
The Target



First Things First

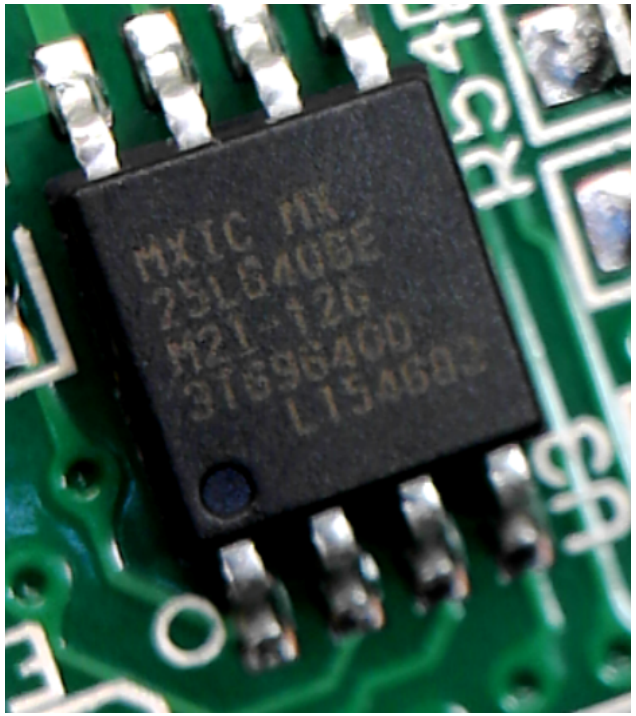


The flip side



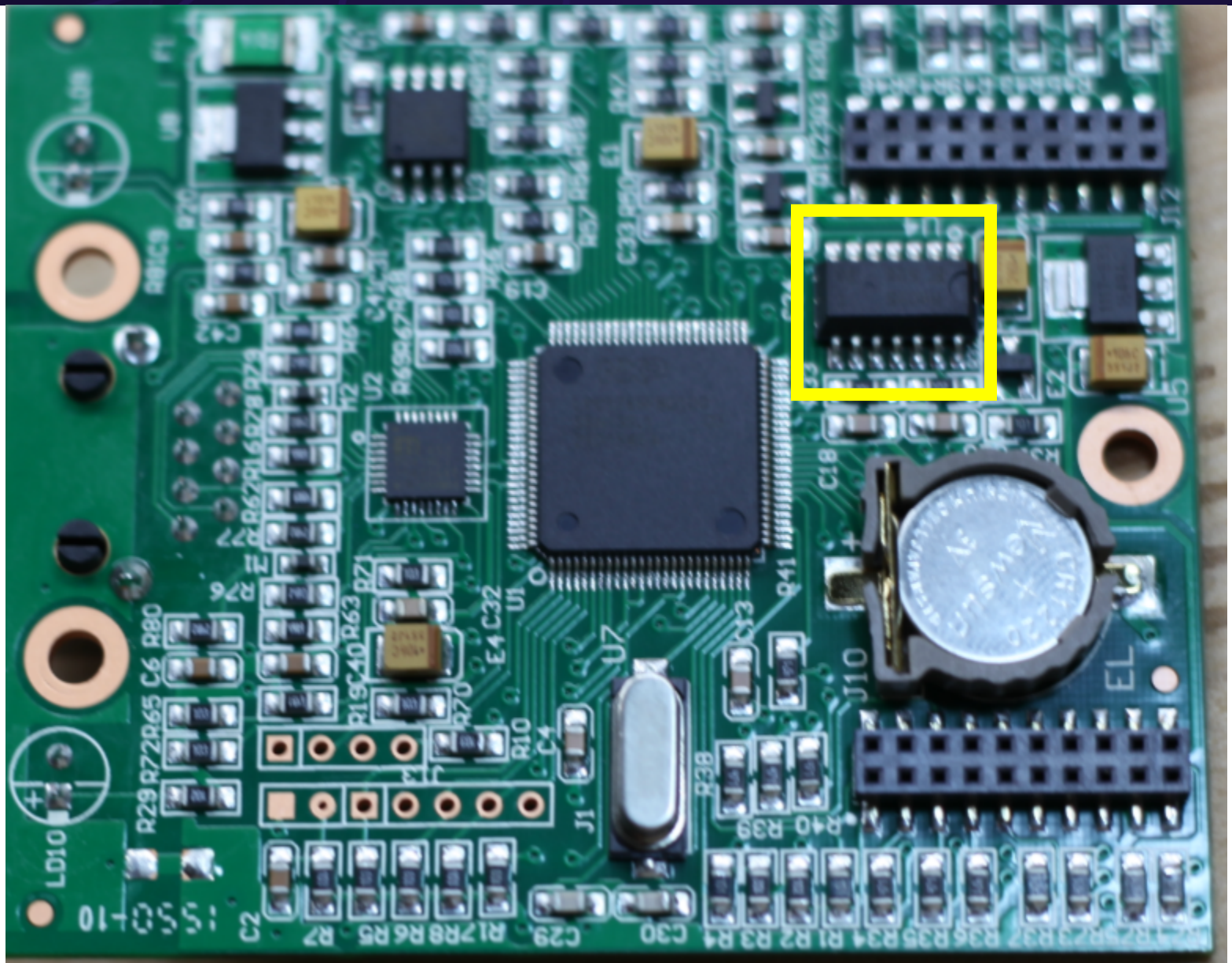
Chip of Interest

- Macronix International 25L6406E
- NOR Flash
- 64 Megabit capacity
- Verdict: Interesting



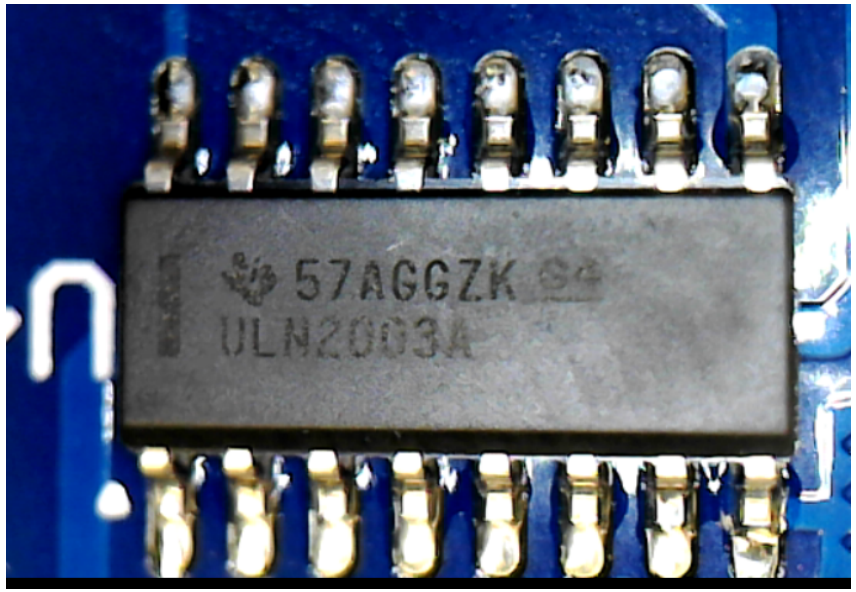
CORE SECURITY

The flip side



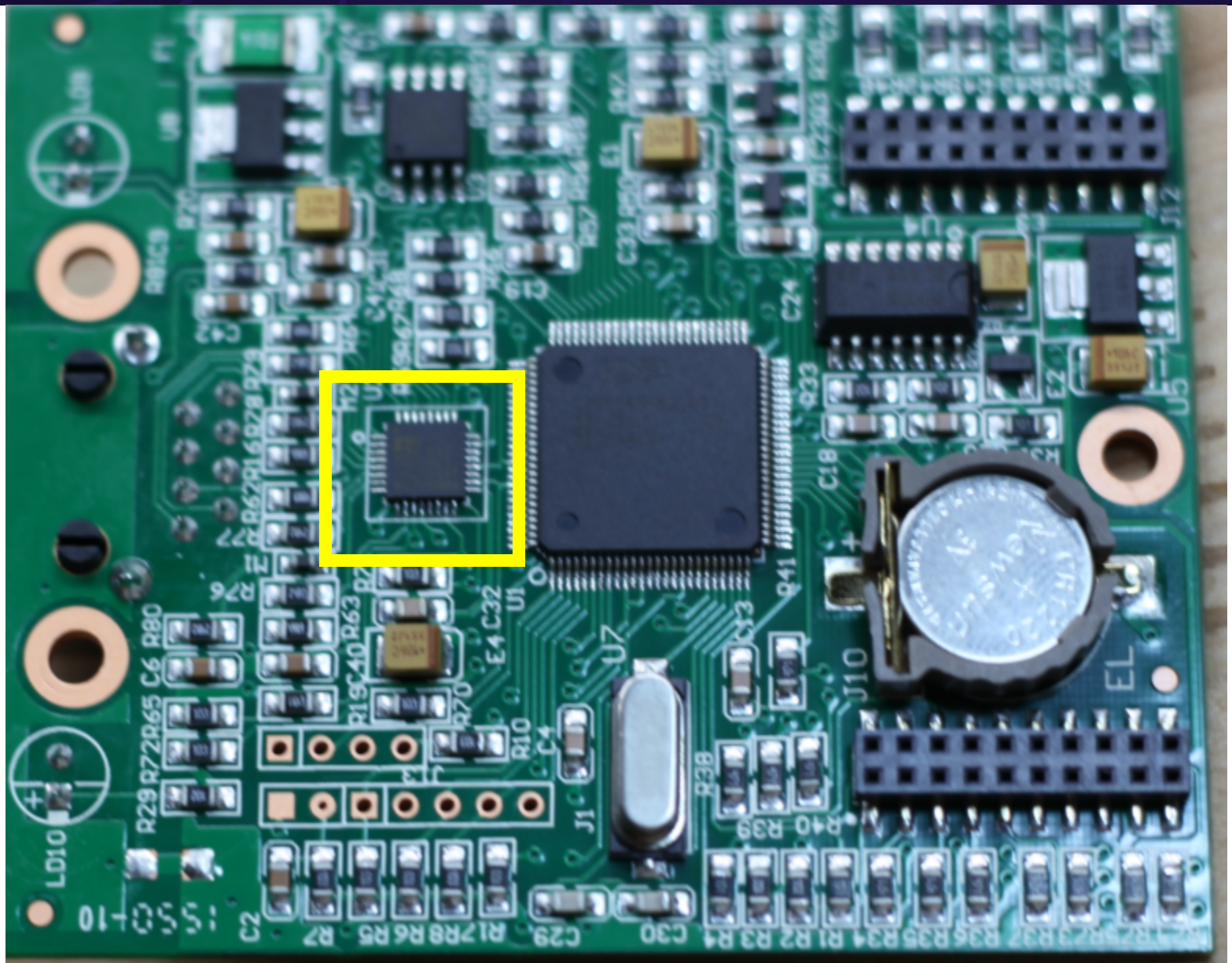
Chip of Interest

- Texas Instruments ULN2003A
- Acts as a relay driver
- BORING



CORE SECURITY

The flip side

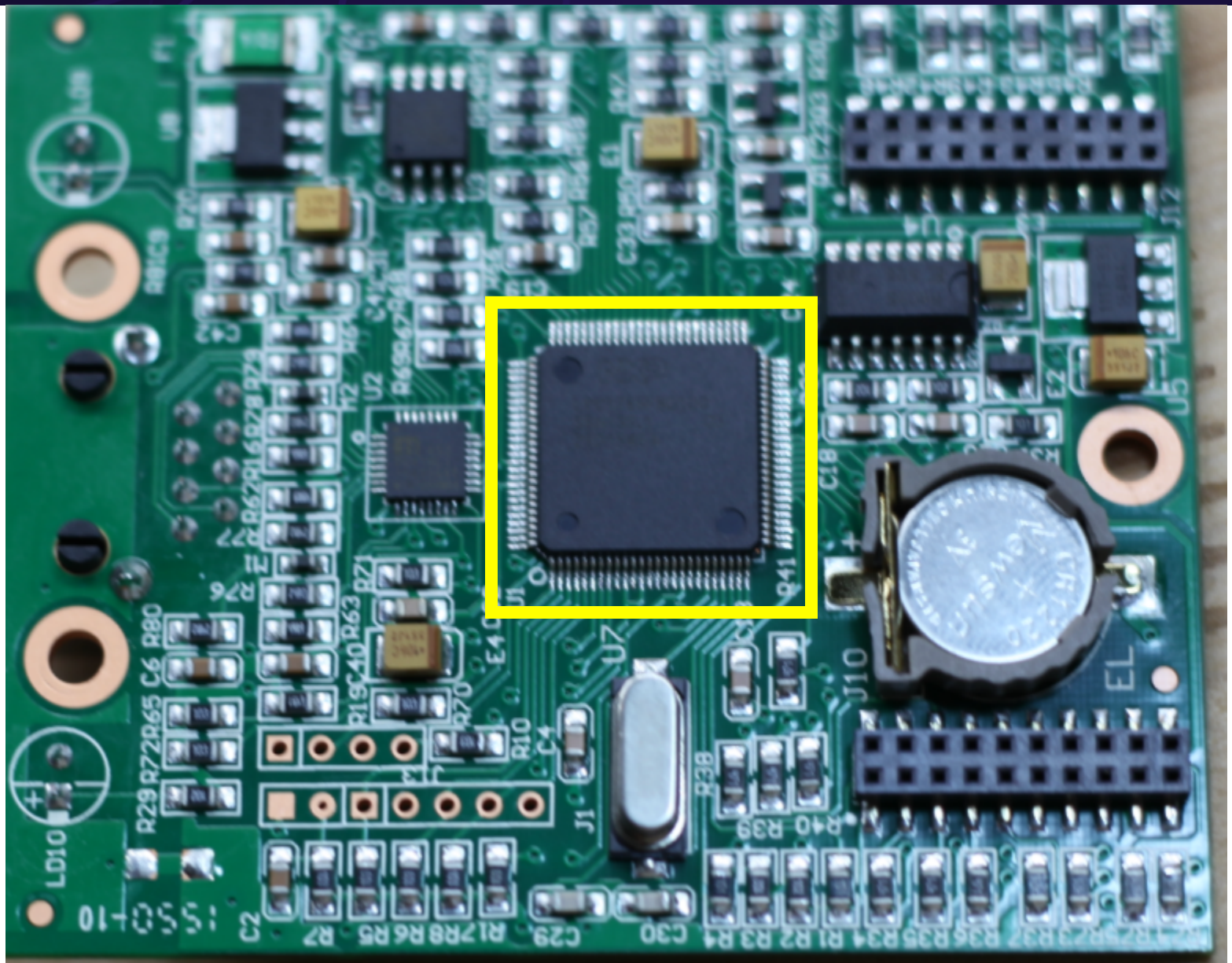


Chip of Interest

- Micrel 8041NL1
- Physical Ethernet Transceiver
- BORING

CORE SECURITY

The flip side

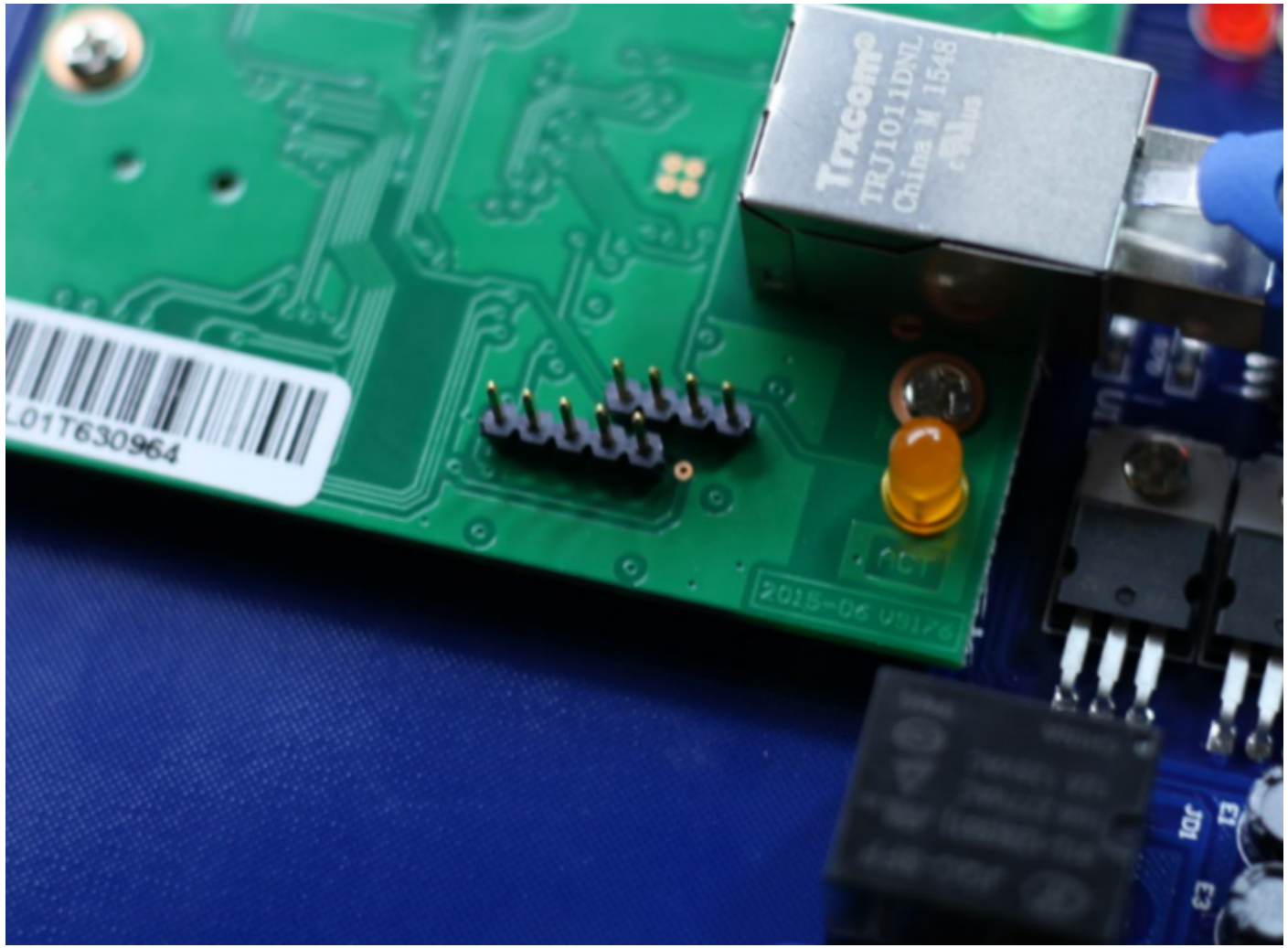


Chip of Interest

- NXP LPC1766FBD100
- ARM based microcontroller
- 256KB onboard flash
- Verdict: Interesting

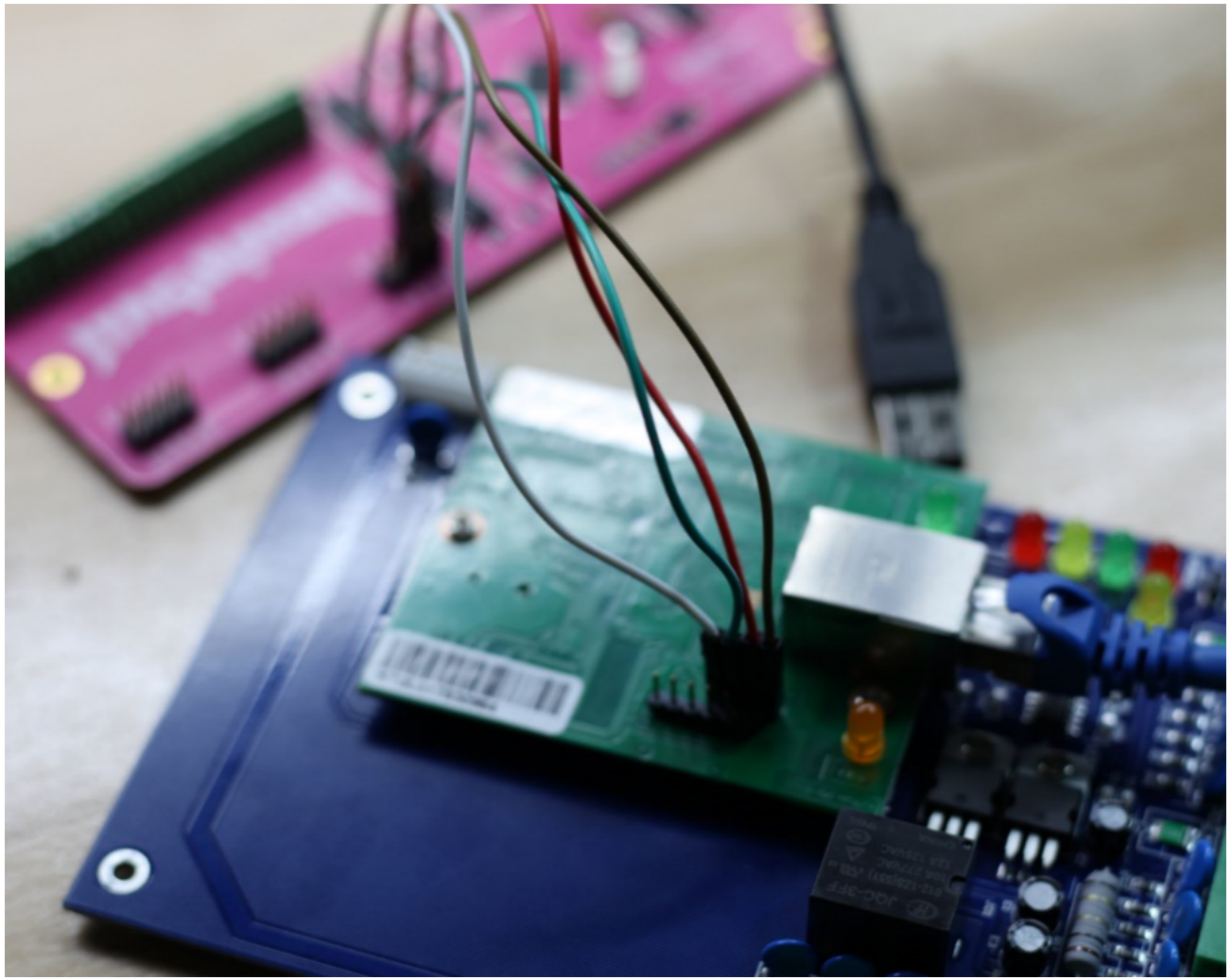
CORE SECURITY

About those through-holes



CORE SECURITY

JTAGULATOR: Activate



CORE SECURITY

JTAGULATOR: Fail

- The JTAGULATOR found a UART, but no JTAG
- Continuity checks on the board showed that JTAG pins were routed to TDI, TMS, TCK, and TDO on the processor.

CORE SECURITY

Frack

8.30.3 Code security (Code Read Protection - CRP)

This feature of the LPC17xx allows user to enable different levels of security in the system so that access to the on-chip flash and use of the JTAG and ISP can be restricted. When needed, CRP is invoked by programming a specific pattern into a dedicated flash location. IAP commands are not affected by the CRP.

There are three levels of the Code Read Protection.

CRP1 disables access to chip via the JTAG and allows partial flash update (excluding flash sector 0) using a limited set of the ISP commands. This mode is useful when CRP is required and flash field updates are needed but all sectors can not be erased.

CRP2 disables access to chip via the JTAG and only allows full flash erase and update using a reduced set of the ISP commands.

Running an application with level CRP3 selected fully disables any access to chip via the JTAG pins and the ISP. This mode effectively disables ISP override using P2[10] pin, too. It is up to the user's application to provide (if needed) flash update mechanism using IAP calls or call reinvoke ISP command to enable flash update via UART0.

CORE SECURITY

Scoreboard

Bobby: 0

Engineers: 1

CORE SECURITY

Let’s try the UART

- No bootup scroll
- Logic analyzer showed random characters
- Meh.

CORE SECURITY

Scoreboard

Bobby: 0

Engineers: 2

CORE SECURITY

How about that Flash chip

- Probed with Logic Analyzer during startup
- No activity at ALL
- Theory: Flash chip is used for storing activity logs

CORE SECURITY

Scoreboard

Bobby: 0

Engineers: 3

CORE SECURITY

Management software

- Installed into a Windows 7 VM
- Communications observed
- .NET Application
- Access Database backend

CORE SECURITY

Communications

- Port UDP/60000
- No cleartext
- Dig deeper

CORE SECURITY

Let’s dig into the Management Software

Login Functionality

```
private void btnOK_Click(object sender, EventArgs e)
{
    if (icOperator.checkSoftwareRegister() < 0)
    {
        using (dfrmRegister dfrmRegister = new dfrmRegister())
        {
            dfrmRegister.Text = CommonStr.strLicenseExpired;
            if (dfrmRegister.ShowDialog(this) != DialogResult.OK)
            {
                wgAppConfig.IsAutoLogin = false;
                return;
            }
        }
    }
    if (icOperator.login(this.txtOperatorName.Text, this.txtPassword.Text))
    {
```

CORE SECURITY

Login Functionality

Oopsie…

```
if (flag && name == "wiegand" && pwd == "168668")
{
    icOperator.m_OperatorID = 1;
    icOperator.m_OperatorName = name;
    result = true;
}
```

CORE SECURITY

Scoreboard

Bobby: 1

Engineers: 3

CORE SECURITY

Password Hashing... not

```
public static string Ept4Database(string StrInput)
{
    string result = "";
    try
    {
        if (Program.Key4Database == null)
        {
            IntPtr intPtr = Marshal.AllocHGlobal(16);
            IntPtr intPtr2 = Marshal.AllocHGlobal(16);
            Program.getKDB(intPtr);
            Program.getIVDB(intPtr2);
            Program.Key4Database = new byte[16];
            Marshal.Copy(intPtr, Program.Key4Database, 0, 16);
            Program.IV4Database = new byte[16];
            Marshal.Copy(intPtr2, Program.IV4Database, 0, 16);
            Marshal.FreeHGlobal(intPtr);
            Marshal.FreeHGlobal(intPtr2);
        }
        byte[] bytes = Encoding.Default.GetBytes(wgTools.SetObjToStr(StrInput));
        using (MemoryStream memoryStream = new MemoryStream())
        {
            using (RijndaelManaged rijndaelManaged = new RijndaelManaged())
            {
                CryptoStream cryptoStream = new CryptoStream(memoryStream, rijndaelManaged.CreateEncryptor(Program.Key4Database, Program.IV4Database), CryptoStreamMode.Write);
                cryptoStream.Write(bytes, 0, bytes.Length);
                cryptoStream.FlushFinalBlock();
                result = Convert.ToBase64String(memoryStream.ToArray());
            }
        }
    }
    catch
    {
        throw;
    }
    return result;
}
```

CORE SECURITY

Oh where, oh where can key be?

```
[DllImport("n3k_comm.dll", CallingConvention = CallingConvention.Cdecl, CharSet = CharSet.Auto)]
public static extern int getKDB(IntPtr k);
```

CORE SECURITY

A novel key generation strategy

```
getKDB:
10011920  push    ebp
10011921  mov     ebp, esp {var_4}
10011923  push    ecx
10011924  mov     dword [ebp-0x4 {var_8}], 0x0
1001192b  jmp     0x10011936

10011936  cmp     dword [ebp-0x4], 0x10
1001193a  jge     0x10011957

10011957  mov     eax, 0x1
1001195c  mov     esp, ebp
1001195e  pop     ebp
1001195f  retn

1001193c  mov     ecx, dword [ebp-0x4 {var_8}]
1001193f  movzx   edx, byte [ecx+0x100b633c]
10011946  mov     eax, dword [ebp-0x4 {var_8}]
10011949  lea     ecx, [edx+eax+0x1]
1001194d  mov     edx, dword [ebp+0x8 {arg_4}]
10011950  add     edx, dword [ebp-0x4 {var_8}]
10011953  mov     byte [edx], cl
10011955  jmp     0x1001192d

1001192d  mov     eax, dword [ebp-0x4 {var_8}]
10011930  add     eax, 0x1
10011933  mov     dword [ebp-0x4 {var_8}], eax
```

CORE SECURITY

Except…

```
IntPtr intPtr = Marshal.AllocHGlobal(16);
IntPtr intPtr2 = Marshal.AllocHGlobal(16);
Program.getKDB(intPtr);
Program.getIVDB(intPtr2);
Program.Key4Database = new byte[16];
Marshal.Copy(intPtr, Program.Key4Database, 0, 16);
Program.IV4Database = new byte[16];
Marshal.Copy(intPtr2, Program.IV4Database, 0, 16);
Marshal.FreeHGlobal(intPtr);
Marshal.FreeHGlobal(intPtr2);
```

CORE SECURITY

Static Keys

- The same values are always passed into the getKDB, so we always get the same key

0000000000000000000

CORE SECURITY

Scoreboard

Bobby: 2

Engineers: 3

CORE SECURITY

Communications Packet

```
public new byte[] ToBytes(ushort srcPort)
{
    byte[] array = new byte[24];
    array[0] = base.type;
    array[1] = base.code;
    Array.Copy(BitConverter.GetBytes(srcPort), 0, array, 2, 2);
    Array.Copy(BitConverter.GetBytes(this._xid), 0, array, 4, 4);
    Array.Copy(BitConverter.GetBytes(base.iDevSnFrom), 0, array, 8, 4);
    Array.Copy(BitConverter.GetBytes(base.iDevSnTo), 0, array, 12, 4);
    array[16] = base.iCallReturn;
    array[17] = this.driverVer;
    array[18] = (byte)wgTools.gPTC_internal;
    array[19] = this.reserved19;
    Array.Copy(BitConverter.GetBytes(this.m_swipeIndex), 0, array, 20, 4);
    ushort value = wgCRC.CRC_16_IBM_CSharp(24u, array);
    Array.Copy(BitConverter.GetBytes(value), 0, array, 2, 2);
    base.EncWGPacket(ref array, array.Length);
    return array;
```

CORE SECURITY

Implications

- No crypto on communications, despite there being functions exported in the unmanaged assembly
- Serial Number used for ID only
- Serial Number is discoverable by a broadcast UDP

CORE SECURITY

Scoreboard

Bobby: 3

Engineers: 3

CORE SECURITY

So what can we do?

Replay attacks if we have pcaps

Changing default lock/unlock on doors

Adding a new prox card

CORE SECURITY

Postmortem

- Hardware implementation reasonably secure
- Software… not so much
- Don’t hardcode backdoors
- Actually Hash your passwords
- Use strong authentication
- Check your crypto

CORE SECURITY

And now…

Continue the discussion

bkuzma@coresecurity.com

@BobbyAtCore

http://www.coresecurity.com

CORE SECURITY