Implementing Practical Electrical Glitching Attacks

November 2015
About Me

Brett Giller

• Computer Security Consultant at NCC group
• Interest in hardware attacks and hardware security
What are glitching attacks

Glitching Attack
• A glitching attack is an intentional fault introduced to undermine device security
• Some of the things the faults can cause
  • Instruction skipping
  • Malformed data reads/write backs
  • Instruction decoding errors
What are glitching attacks

Noninvasive
- Involves minimal damaging to IC packaging
- Can be done on the cheap
- Relatively simple to implement

Invasive/Semi-Invasive
- Required to decapsulate/modify IC packaging
- Needs fairly expensive equipment
- Potentially a larger amount of time required
What are glitching attacks

Noninvasive types

- Electrical
- Clock
- Power
- Thermal
- Radiation
What are glitching attacks

Clock
- Introduce unplanned clock edge(s) to device
- Different glitch signals can be used
  - 3 Phase Xor (See “Gliching for n00bs” by Exide)
  - Direct Xor Duration
  - Increased clock speed
What are glitching attacks

Power
• Pull to ground (brownout)
  • More likely to cause certain instructions to fail
  • More predictable
  • Causes prorogation delays in IC
• Increase voltage (spiking)
  • Easy to implement
  • Also easy to damage target
  • Likely adds more floating signals
Where can we use glitching?

Targets
• Game consoles
• Copy protected IC
• Door locks/Safes
• Set top boxes
• Mobile hotspots

Code
• Authentication checks
• Bounds/sanity checks
• Memory read/writes
Xbox 360 Reset Glitch Hack

• Original exploit and write up by GliGli
• Attack similar to clock glitching attack
• Takes advantage of exposed IC pin interface
• Generally a non patchable exploit
• Only describing exploit on original xbox 360
Xbox 360 Reset Glitch Hack

Xbox 360 Bootloader Security

• Works by chain of bootloaders starting with ROM (1BL) then subsequently loading hypervisor/base kernel, the kernel and then the dash
• Nand code is RSA signed
• Xbox 360 emits out diagnostic signals during this process
Xbox 360 Reset Glitch Hack

Vulnerable Pin Interfaces
• CPU_PLL_BYPASS
• RESET behavior which skips instruction execution
• Diagnostic POST bus
Xbox 360 Reset Glitch Hack

CPU_PLL_BYPASS
• CPU_PLL_BYPASS slows processor to 520 khz
• Allows for easy timing
Xbox 360 Reset Glitch Hack

RESET
• CPU Reset is a common feature on most ICs which typically resets execution state
• Processor instead effectively skips instructions when RESET is asserted
Xbox 360 Reset Glitch Hack

POST bus
• POST bus is an 8-bit diagnostic bus
• Emits signals for important steps of the booting process
• Sends 0x36 when decrypting CD and 0x39 when comparing the hash
Xbox 360 Reset Glitch Hack

What the exploit targets
• Target’s signature check
• Use POST bus signals to know when to do so

About the payload
• Payload is set up in a way that resets the xbox when it fails
• Indicator about the nature of glitching
Xbox 360 Reset Glitch Hack

Exploit
1. Upload payload to NAND
2. Monitor for POST 0x36 (decrypting the base kernel)
3. Assert CPU_PLL_BYPASS
4. Monitor for POST 0x39
5. Start internal counter to end at ~62% of the POST 0x39 length
6. Assert RESET for 100 ns at end of counter duration
7. Resume normal execution
Xbox 360 Reset Glitch Hack

Vendor Response
• Dash version 7371 destroyed JTAG fuses
• Additional check added to loading of base kernel
• Began banning users from Xbox Live
Xbox 360 Reset Glitch Hack

Reset Glitch 2.0
• Exploit said to work on all original xbox 360s
• Makes use of i2C bus on xbox 360 to slow down clock
• Made use of NAND “DemoN” to avoid getting Xbox Live bans
Xbox 360 Reset Glitch Hack

1. Wait for POST signal 36
2. Assert CPU_PLL_BYPASS
3. Monitor for POST Signal 39
4. Increment Counter
5. Start Counter
6. Counter Reaches magic value
7. Assert Reset pin for 100ns
8. Resume normal execution
Attack Methodology

General Methodology
• Assess target device for target code
• Review datasheet of target IC
• Test target threshold manually
• Find points to attack on target device
• Search for signals on target device
• Prepare target for glitching
• Setup FPGA for brute forcing of glitching parameters
• Begin attack
Attack Methodology

Choosing target code
- Search for checks which use instructions longer than one cycle
- Instructions with write back
- Security checks near computationally intense code
- Easy/Quick to reset state upon failure
- Code near boot sequence or near kernel operations
Attack Methodology

Reviewing Datasheets
• Operational ranges
• Brownout detection
• Security features (if any)
• Rated clock speeds
• Interesting pin interfaces
Attack Methodology

Testing target threshold manually
• Useful if the target device has a development board
• Test brownout threshold
• Duty cycle testing of device
• View delay ranges of example instructions in practice
Attack Methodology

Findings points to attack on a device
• External crystals
• Decoupling capacitors
• Voltage regulators
• Voltage dividers
• Power supply
Attack Methodology

Signals
• Signals are information emitted from the device which allows us to narrow the values we have to guess
• They vary greatly in terms of value due to noise introduced during execution
• The parameter we are guessing here is the delay range maximum
• Signals can be used to start the delay state or to close in closer to the signal which does that
Attack Methodology

What a signal can be
• GPIO toggling
• Status LEDs
• Serial Messages
• Device specific diagnostic buses
• Power analysis
• Calculation of instruction timings from input
• Raw timing window
Attack Methodology

Calculating timing from disassembly

- Signal delays can sometimes be calculated to a tight range
- This requires the target code to be writing out somewhere within close proximity
- During the experiments the following equation had a generally good result

\[
\left[ \frac{i_{\text{instCount}} T_{\text{target}}}{T_{\text{Glitch}}} - 3P_{\text{size}}, \frac{i_{\text{cycleCount}} T_{\text{target}}}{T_{\text{Glitch}}} + 3P_{\text{size}} \right]
\]
Attack Methodology

Preparing the device
- What is desoldered or not is dependent on the type of glitch used
- Clock glitching involves removal of the device crystal and its 2 nearby decoupling capacitors
- Voltage spiking involves finding a VCC decoupling capacitor and soldering to the high side of the capacitor
- Voltage brownout attacks work best by removing all of the power decoupling capacitors and then either cutting the line on the PCB or finding a nearby voltage regulator
Attack Methodology

1. Wait for ignition signal
2. Send test case out or get initial signal for starting delay
3. Calculate value from experimental trials
4. Delay State
5. Choose to see if successful or change parameters
6. If successful, communicate out resultant values
7. Change glitch duration or timing delay
8. Depending on target, it may be reset
9. Raw timing value calculated from heuristic
10. Optional wait period for checking signal results
Attack Methodology

Glitching parameters

- Glitch duration
  - Typically maxed out by datasheet values or experimental values
  - Suggest for clock glitching signal to at least cover one full cycle
- Delay length
  - Determined by signals or by calculations from disassembly
  - Parameter with the widest range
- Glitch Attempts
  - Glitching is not always accurate, and can often require more than one try to work
  - Targeting a state which can be quickly recovered greatly accelerates process
Attack Methodology

Possible FPGA Glitching Components

- **Glitching device**
  - Defined as device which emits out the supplied glitch signal
  - Should not be changed very often

- **Test case device**
  - Optional device which emits out test case to the target device
  - Emits signal back to the glitching device of when the test case is done

- **Sampling device**
  - Device which signals back to the glitching device whether the glitch was successful
  - Condition for tests of whether attack is successful should have highest precedence
Tools used

- Used DE0 Nano FPGA device
- When choosing a FPGA dev board for attacks consider its speed
- Attacks implemented used multiple GPIO pins
- Also used switching transistors for trying attacks on 5V logic devices
- Consider output voltage and maximum clock speed when choosing an FPGA
ATMega328p

Attacks Used
• Clock Glitching
  • Got fairly consistent results from it
• VCC Brownout

Code Targeted
• Simple code to glitch out of extremely long loop
• Used 2 devices which had different delay values in them
ATMega328p

- Was used as a starting example
- Both glitching attacks were direct input to the device
- Used standard breadboard setup
- Reset target device during non response
- Gained fairly consistent results on both attacks for values found
- Performed attacks where FPGA directly supplied what was being glitched
ATMega328p

Signal used
• Used timing of target toggling LED on and off

Threshold testing
• To test brownout limit, we employ the simple state machine from before slowly incrementing the power off until the example code turns the LED off
• During duty cycle testing I found that it runs off of very small duty cycles of ~8%
ATMega328p

Threshold Testing results

• Datasheet claims that the timeout period was 2 us
• The limit was found to be 200ns in practice
ATMega328p

Instruction Counting in practice
- When a direct count according to the disassembly was tried, the attack failed
- Instead when a range was used the attack succeeded
- The actual value for the delays were smaller than expected
- Used signal to help locate when to start counter
- P_size -> pipeline size, T_* -> period
- Rough heuristic equation used to brute force delay value from ideal signal

\[
\left[ \frac{i_{\text{instCount}} T_{\text{target}}}{T_{\text{Glitch}}} - 3P_{\text{size}}, \frac{i_{\text{cycleCount}} T_{\text{target}}}{T_{\text{Glitch}}} + 3P_{\text{size}} \right]
\]
ATMega328p Clock Setup
ATMega328p

Clock Glitching
• Used duration of 60-80 ns for glitch pulse
• Directly xored against 16MHz clock supplied
• Found fairly consistent timing delays
• Value found was smaller than expected
ATMega328p Voltage Setup

About
• The ATMega328p can run off of both 3.3v and 5v making this easier
ATMega328p

Voltage Glitching
- Performed Brownout glitching
- Had fairly similar timings
- Durations varied for successful attempts
- Generally took more tries to work than the clock glitching setup
- If possible directly power target device
ATMega328p

FPGA Parts
- Glitch device
  - Used direct xor, and 3 phase xor successfully
  - Maxed out brownout timing to be the maximum tested threshold
- Test Case
  - Waited for LED to turn on then off again
  - Helped gain a smaller window to timing
- Sampler
  - Direct connection to LED which turns on after a successful branch skip
  - Checked at the top level of the glitch device code
ATMega328p

Problems Encountered
• FPGA IDE issues
• Logic translation between ATMega328p and FPGA
• Analog problems
• Extra delay introduced from transistor switch
LPC1343

What it is
- ARM development board from programmable logic training class
- Communicates out over serial asking for password
- Meant to be a timing attack example
- Completely blackbox example
LPC1343

Signal used
• Used response from serial to determine the timing window
• Also monitored for “I” in message to determine if the response was successful or not
• Required FPGA to send input to device before glitching
• Signal had large ranges in practice
• Needed a post glitch wait phase to allow parsing of serial input to determine if a good value has been found
LPC1343

Preparing the board
• Removed crystal on board with a bit of heat and mechanical force
• Also removed decoupling capacitors
ATMega328p

FPGA Parts
- Glitch device
  - Only attempted clock glitching
  - Virtually the same verilog code to the previous example
- Test Case
  - Sent out 17 “A”s to the target device over serial
- Sampler
  - Monitor to check for incorrect password message
  - If the message does not match send back the parameters used over serial
LPC1343

Conclusions

• Serial interface messages are alright to use as a signal
• Finding the correct glitching parameters black box is fairly difficult
• Glitching parameters vary depending on the situation and device architecture
Example Enumeration - Door Lock
Example Enumeration - Intel Galileo
Example Enumeration - Set top box
Example Enumeration - Set top box
Experiments for the future

- Clock glitching on a device with a frequency multiplier
- Attacking devices which claim to have mitigations
- Combining multiple types of glitching attacks
- Attacking memory devices
- Implementing an attack on code running on top of an operating system
Defenses Against Glitching

• Glitch detectors
• High quality brownout detection
• Lockstep cores performing checks on one another
• Asynchronous internal clock with dummy cycles
• Internal Oscillators
• Halt on invalid instruction execution
• Lock down unnecessary diagnostic signals
Defenses Against Glitching

• Search for ICs which use mitigations against glitching
• Perform assessments against the IC before using it in production
• Writing code defensive in the case of a glitching attack would only buy time
Conclusions

• Glitching attacks are cheap, though can vary in implementation.
• When performing a glitching attack, multiple tries are often required.
• Try before you buy if you are an embedded system vendor.
References / Suggested Reading

• Glitching for Noobs - Exide, RECON 2014
• Glitching and Side Channel Analysis for all, Colin O’Flynn RECON 2015
• Copy Protection in Modern Microcontrollers, Sergei P. Skorobogatov
• https://gbatemp.net/threads/x360-the-reset-glitch-hack.306685/
• http://www.newae.com/sidechannel/cwdocs/naecw1173_cwlite.html
• Modern Game Console Exploitation, DeBusschere and McCambridge
References / Suggested Reading

• http://rdist.root.org/2010/01/27/how-the-ps3-hypervisor-was-hacked/
• http://beta.ivc.no/wiki/index.php/Xbox_360_Kernel
• https://eprint.iacr.org/2004/100.pdf
• https://rgsilva.com/Bachelorarbeit.pdf
• https://wiki.crypto.rub.de/summerschool/slides/1_vanwoudenber.pdf
• http://www.t4f.org/articles/fault-injection-attacks-clock-glitching-tutorial/
Point of contact

Brett Giller
Security Consultant

E: brett.giller@nccgroup.trust
Github: breadBurglar
## Locations

<table>
<thead>
<tr>
<th>North America</th>
<th>Europe</th>
<th>Australia</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlanta</td>
<td>Manchester - Head Office</td>
<td>Sydney</td>
</tr>
<tr>
<td>Austin</td>
<td>Amsterdam</td>
<td></td>
</tr>
<tr>
<td>Chicago</td>
<td>Basingstoke</td>
<td></td>
</tr>
<tr>
<td>Kitchener</td>
<td>Cambridge</td>
<td></td>
</tr>
<tr>
<td>New York</td>
<td>Copenhagen</td>
<td></td>
</tr>
<tr>
<td>San Francisco</td>
<td>Cheltenham</td>
<td></td>
</tr>
<tr>
<td>Seattle</td>
<td>Edinburgh</td>
<td></td>
</tr>
<tr>
<td>Sunnyvale</td>
<td>Glasgow</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Leatherhead</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Leeds</td>
<td></td>
</tr>
<tr>
<td></td>
<td>London</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Luxembourg</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Malmö</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Milton Keynes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Munich</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Vilnius</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wetherby</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zurich</td>
<td></td>
</tr>
</tbody>
</table>
Agenda

What are electrical glitching attacks
Where are they applicable
Reset Glitch Hack on the Xbox 360
Methodology for performing an attack
Tools used
Atmega328p
LPC1343
Example Enumeration
Glitching experiments for the future
Defenses against glitching