ICSPatch: Automated Vulnerability Localization and Non-Intrusive Hotpatching in Industrial Control Systems using Data Dependence Graphs

Prashant Rajput    Constantine Doumanidis    Michail Maniatakos
What are Industrial Control Systems?

• Industrial Control Systems (ICS)
  • Ruggedized systems
  • Interface with the real world
  • Examples: PLCs, SCADA systems, etc.
What are Industrial Control Systems?

• Industrial Control Systems (ICS)
  • Ruggedized systems
  • Interface with the real world
  • Examples: PLCs, SCADA systems, etc.

• Part of critical infrastructure
  • Power grid
  • Nuclear plants
  • Desalination facilities
What are Industrial Control Systems?

- Industrial Control Systems (ICS)
  - Ruggedized systems
  - Interface with the real world
  - Examples: PLCs, SCADA systems, etc.

- Part of critical infrastructure
  - Power grid
  - Nuclear plants
  - Desalination facilities

- ICS robustness is paramount for safety
What are Modern ICS?

- Industry 4.0 / Industrial IoT
- ICS evolve into typical computers
  - Generic third-party SoCs
  - General-purpose OS
  - Remote connections

PLC Execution Model

• Programmable Logic Controllers (PLCs)
  • An industrial computer continuously monitoring the state of input, makes decision based on a custom program to control state of output devices

Fig. 1: Execution model for Codesys runtime.
PLC Execution Model

- Programmable Logic Controllers (PLCs)
  - An industrial computer continuously monitoring the state of input, makes decision based on a custom program to control state of output devices

- Runtime
  - Collection of components necessary for proper execution of the application binary

Fig. 1: Execution model for Codesys runtime.
PLC Execution Model

- Programmable Logic Controllers (PLCs)
  - An industrial computer continuously monitoring the state of input, makes decision based on a custom program to control state of output devices

- Runtime
  - Collection of components necessary for proper execution of the application binary

- Scan Cycle
  - Continuously scan program, input scan, execute program, output scan

Fig. 1: Execution model for Codesys runtime.
PLC Execution Model

• Programmable Logic Controllers (PLCs)
  • An industrial computer continuously monitoring the state of input, makes decision based on a custom program to control state of output devices

• Runtime
  • Collection of components necessary for proper execution of the application binary

• Scan Cycle
  • Continuously scan program, input scan, execute program, output scan

• Control Application
  • IEC 61131-3 compliant code regulating a physical industrial process

Fig. 1: Execution model for Codesys runtime.
PLC Binary Crashes!

- Crashes are signals of potentially exploitable vulnerabilities
- Vulnerabilities need to be patched
- Patching requires:
  - Vendor to produce a patch
  - PLC to be restarted
- However:
  - Vendors may not be able to produce a patch quickly (or ever)
  - PLC cannot be taken offline before next scheduled downtime
ICSPatch

- Hotpatching
  - Dynamically updating application without downloading a new version or even restarting it
ICSPatch

• Hotpatching
  • Dynamically updating application without downloading a new version or even restarting it

• Why?
  • Hotpatching for real-time applications remains unexplored, except HERA [1] and RapidPatch [2]
  • Here, application binary executes in the context of a runtime
  • Proprietary format
  • Unknown vulnerabilities
  • No upstream patch source

Fig. 1: Codesys-based PLC software stack.

ICSPatch

- Creating a diverse dataset
  - 5 sectors
  - 4 type of vulnerabilities
    - OOB write
    - OOB read
    - OS command injection
    - Improper input validation
  - 4/5 most dangerous software weaknesses for 2021 [1]

Table 1: A diverse synthetic control application dataset.

<table>
<thead>
<tr>
<th>Shared Library</th>
<th>Imported Functions</th>
<th>Aircraft Flight Control</th>
<th>Anaerobic Digestion Reactor</th>
<th>Chemical Plant</th>
<th>Desalination Plant</th>
<th>Smart Grid</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysMem23</td>
<td>SysMemSet</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>SysMemMove</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>SysMem</td>
<td>SysMemSet</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>SysMemMove</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>SysMemCpy</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>MemUtils</td>
<td>MemSet</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>MemCpy</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>BitCpy</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>IEC 61131-3 code</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Out-of-bounds array index</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

CWE-787/CWE-125: Out-of-Bounds Write/Read  
CWE-78: OS Command Injection  
CWE-20: Improper Input Validation

Methodology

System design overview

**Threat Model**
- Remote adversary with MiTM capabilities
- Adversary limited to data injection/modification attacks
- ICSPatch does not assume upstream patch source
- ICSPatch assumes at least one exploit input
Methodology

System design overview

- Threat Model
  - Remote adversary with MiTM capabilities
  - Adversary limited to data injection/modification attacks
  - ICSPatch does not assume upstream patch source
  - ICSPatch assumes at least one exploit input
Methodology

Step 1: Vulnerability Identification & Localization

- **Vulnerability Identification**


<table>
<thead>
<tr>
<th>Name</th>
<th>Action</th>
<th>Patch Identifier</th>
<th>Definition</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>OOB_READ</td>
<td>READ_ADDRESS &gt; RUNTIME_STACK AND READ_ADDRESS &lt; RUNTIME_DATA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS_CMD_INJ</td>
<td>WRITE_ADDRESS &gt; RUNTIME_STACK AND WRITE_ADDRESS &gt; RUNTIME_ADDRESS_TABLE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 1: ICSPatch System Design.

Fig. 2: ICSPatch Rule Example.
Methodology

Step 1: Vulnerability Identification & Localization

- **Vulnerability Localization**
  - Traverse back on the DDG
  - Locate the closest node to the boundary between control application and the runtime.

Fig. 1: ICSPatch System Design.

Fig. 3: Vulnerability localization in ICSPatch using Data Dependence Graph.
Methodology

Step 1: Vulnerability Identification & Localization

- **Vulnerability Localization**
  - Traverse back on the DDG
  - Locate the closest node to the boundary between control application and the runtime.
Methodology

Step 1: Vulnerability Identification & Localization

- **Vulnerability Localization**
  - Traverse back on the DDG
  - Locate the closest node to the boundary between control application and the runtime.
Methodology

Step 2: Patch Generation & Deployment

- **Patch Generation**
  - No upstream patch sources for control application
  - Memory related vulnerabilities require bound checking patches
  - Populate skeleton patches with:
    - Vulnerable bound memory location
    - User defined bound
    - Next function offset into the address table
Methodology

Step 2: Patch Generation & Deployment

**Patch Generation**
- No upstream patch sources for control application
- Memory related vulnerabilities require bound checking patches
- Populate skeleton patches with:
  - Vulnerable bound memory location
  - User defined bound
  - Next function offset into the address table

**Patch Verification**
- Load the patch in angr simulation instance
- Execute and check vulnerability rulesets
Methodology

Step 2: Patch Generation & Deployment

- **Branching in Control Applications**
  1. Load base address of address table

Fig. 1: ICSPatch System Design.

Fig. 4: Branching in Codesys compiled control applications.
Methodology

Step 2: Patch Generation & Deployment

- **Branching in Control Applications**
  1. Load base address of address table
  2. Load the address of the next function
Methodology

Step 2: Patch Generation & Deployment

- **Branching in Control Applications**
  1. Load base address of address table
  2. Load the address of the next function
  3. Modify the value of the PC

---

**Fig. 1: ICSPatch System Design.**

**Fig. 4: Branching in Codesys compiled control applications.**
Methodology

Step 2: Patch Generation & Deployment

Patch Deployment
1. Write patch at empty memory location
2. Write patch address into an empty address table entry
3. Modify the offset to the base address table to load patch address (critical)
## Methodology

### Step 2: Patch Generation & Deployment

<table>
<thead>
<tr>
<th>Development PLC</th>
<th>Patch Server [ICSPatch]</th>
<th>Deployed PLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Codesys execution state</td>
<td>Send Information</td>
<td>Get process related information</td>
</tr>
<tr>
<td>• Hexdumps&lt;br&gt;• Codesys runtime&lt;br&gt;• Control application&lt;br&gt;• Shared libraries: Libc, Libm&lt;br&gt;• Process Id</td>
<td></td>
<td>Execute IEC Code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Determine end condition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Select candidate patching location</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Check end condition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Calculate patch related offsets</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Patch Deployment

1. Write patch at empty memory location
2. Write patch address into an empty address table entry
3. Modify the offset to the base address table to load patch address (critical)

---

**Fig. 1:** ICSPatch System Design.

**Fig. 5:** Steps to modify control flow in control applications.

### Methodology

#### Patch Deployment

1. Write patch at empty memory location
2. Write patch address into an empty address table entry
3. Modify the offset to the base address table to load patch address (critical)
## Methodology

### Step 2: Patch Generation & Deployment

<table>
<thead>
<tr>
<th>Development PLC</th>
<th>Patch Server [ICSPatch]</th>
<th>Deployed PLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Codesys execution state</td>
<td>Build DDG</td>
<td>Get process information</td>
</tr>
<tr>
<td>Send Information</td>
<td>Execute IEC Code</td>
<td>Get bound input</td>
</tr>
<tr>
<td>• Hexdumps</td>
<td>Check for Vulnerability</td>
<td>Install patch in simulation and verify safety</td>
</tr>
<tr>
<td>• Codesys runtime</td>
<td>Yes</td>
<td>LKM Patchers</td>
</tr>
<tr>
<td>• Control application</td>
<td>No</td>
<td>Control Application</td>
</tr>
<tr>
<td>• Shared libraries: Libc, Libm</td>
<td>Yes</td>
<td>#</td>
</tr>
<tr>
<td>• Process Id</td>
<td></td>
<td>User Level Kernel Level</td>
</tr>
</tbody>
</table>

#### Patch Deployment

1. Write patch at empty memory location
2. Write patch address into an empty address table entry
3. Modify the offset to the base address table to load patch address (critical)

**Fig. 1: ICSPatch System Design.**

**Fig. 5: Steps to modify control flow in control applications.**

**Methodology**

- **Patch Deployment**
  1. Write patch at empty memory location
  2. Write patch address into an empty address table entry
  3. Modify the offset to the base address table to load patch address (critical)
Methodology

Step 2: Patch Generation & Deployment

- Patch Deployment
  1. Write patch at empty memory location
  2. Write patch address into an empty address table entry
  3. Modify the offset to the base address table to load patch address (critical)
Experimental Results

- **Timing Overhead**
  1. **Normal**: 13 instructions (32 bits) + patch address + hook
  2. **Exception**: Does not load base address and removes ldr instruction
  3. Increased latency due to program structure (loop)
  4. Critical operation modifying execution flow by overwriting ldr offset (hook)
  5. Minimum scan cycle impact

<table>
<thead>
<tr>
<th>Critical Infrastructure</th>
<th>Vulnerability</th>
<th>Vulnerability Localization</th>
<th>Patch Generation</th>
<th>Patch Verification</th>
<th>Mean Execution Time (ms)</th>
<th>Achieved Scan Cycle (ms)</th>
<th>Memory (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aircraft Flight Control</td>
<td>CWE-20</td>
<td>3.08</td>
<td>178.87</td>
<td>7.71</td>
<td>253.18</td>
<td>32.22</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>CWE-787</td>
<td>6.72</td>
<td>166.39</td>
<td>67.18</td>
<td>332.73</td>
<td>18.05</td>
<td>20.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>232.21</td>
<td>19.56</td>
<td>21.26</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>218.51</td>
<td>21.36</td>
<td>21.78</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>203.17</td>
<td>16.61</td>
<td>20.53</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>193.59</td>
<td>15.04</td>
<td>18.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>180.73</td>
<td>13.43</td>
<td>16.70</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>168.51</td>
<td>11.91</td>
<td>15.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>158.73</td>
<td>10.37</td>
<td>13.43</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>150.91</td>
<td>9.01</td>
<td>12.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>143.88</td>
<td>7.73</td>
<td>9.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>137.80</td>
<td>6.39</td>
<td>7.73</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>132.70</td>
<td>5.05</td>
<td>6.39</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128.61</td>
<td>3.71</td>
<td>5.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>125.52</td>
<td>2.41</td>
<td>3.71</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>122.41</td>
<td>1.09</td>
<td>2.41</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>119.31</td>
<td>0.82</td>
<td>1.09</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>116.23</td>
<td>0.54</td>
<td>0.82</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>113.16</td>
<td>0.29</td>
<td>0.54</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>109.98</td>
<td>0.01</td>
<td>0.29</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>106.80</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>103.54</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100.31</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>97.08</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>93.86</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>90.63</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>87.39</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>84.16</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>80.92</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>77.69</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>74.46</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>71.22</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>67.99</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>64.76</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>61.53</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>58.29</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>55.06</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>51.82</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>48.59</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>45.35</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>42.12</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>38.88</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>35.64</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32.40</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>29.17</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25.93</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>22.69</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>19.45</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16.21</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12.97</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9.73</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.49</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.25</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 2: Detailed breakdown of ICSPatch used on Aircraft Flight Control CWE-20 vulnerable binary.
Experimental Results

- Codesys runtime utilizes 14% and 11% CPU for WAGO PFC 100 and 200, respectively
- Before the critical operation
  - Change runtime’s nice value to 19 (lowest)
  - preempt_disable() and local_irq_disable()
Case Study

Fig. 1: hardware-in-the-Loop setup of MSF desalination plant.

- **Experimental Setup**
  - MATLAB Simulink model for a Multi-Stage Flash desalination plant validated against the Khubar II plant in Saudi Arabia
  - NI USB 6002, a DAQ device connects Simulink model to WAGO PFC100 PLC
  - ICSPatch server connects to the PLC
Case Study

- **Experimental Setup**
  - MATLAB Simulink model for a Multi-Stage Flash desalination plant validated against the Khubar II plant in Saudi Arabia
  - NI USB 6002, a DAQ device connects Simulink model to WAGO PFC100 PLC
  - ICSPatch server connects to the PLC
Case Study

- Experimental Setup
  - MATLAB Simulink model for a Multi-Stage Flash desalination plant validated against the Khubar II plant in Saudi Arabia
  - NI USB 6002, a DAQ device connects Simulink model to WAGO PFC100 PLC
  - ICSPatch server connects to the PLC
Thank you. Questions?

@starlordphr
@c_smokeson
@realMoMAlab

https://wp.nyu.edu/momalab/