The Plan

• Administrivia
• Parser sample code
• Break/Fix round
• In-class build time!
Administrivia

- Build round ends Sunday (1/12) at midnight
- Design doc updates due Monday (1/13) at midnight
  - One per team
  - Same format as prior version
- Mid-course survey due Monday at midnight
  - Emails will be sent out today
Parser example code
Break/Fix Round

• Identify bugs in other team’s code
  • You’ll have access to other team’s source
• Fix bugs found in your code
Break Submissions

Three types of breaks:

1. Security:
   - Oracle returns DENIED_*, but the target doesn’t (confidentiality, integrity)
   - Oracles returns correctly, but the target returns DENIED_* (availability)
   - Oracle times out, but the target hangs (availability)
Break Submissions

{
  "type": "integrity",
  "target_team": 9,
  "arguments": {
    "argv": ["%PORT%","password"],
    "base64": false
  },
  "programs": [
    {"program": "as principal admin password \"password\" do\nset x = \"x\"\nreturn x\n***", "base64": false},
    {"program": "as principal admin password \"wrongpassword\"
      do\nreturn x\n***", "base64": false}
  ],
  "configuration": {
    "sensors": {
      "temperature": "80"
    },
    "output_devices": {
      "lights": "0"
    }
  }
}

Break Submissions

```json
{
  "type": "integrity",
  "target_team": 9,
  "arguments": {
    "argv": ["%PORT%","password"],
    "base64": false
  },
  "programs": [
    {
      "program": "as principal admin password "password" do
      \n      x\n      return x
      ***", "base64": false
    },
    {
      "program": "as principal admin password "wrongpassword"
      do
      return x
      ***", "base64": false
    }
  ],
  "configuration": {
    "sensors": {
      "temperature": "80"
    },
    "output_devices": {
      "lights": "0"
    }
  }
}
```
Break Submissions

```json
{
  "type": "integrity",
  "target_team": 9,
  "arguments": {
    "argv": ["%PORT\%", "password"],
    "base64": false
  },
  "programs": [
    {"program": "as principal admin password \"password\" do
\nset x = \"x\n\nreturn x\n***", "base64": false},
    {"program": "as principal admin password \"wrongpassword\"
     do\nreturn x\n***", "base64": false}
  ],
  "configuration": {
    "sensors": {
      "temperature": "80"
    },
    "output_devices": {
      "lights": "0"
    }
  }
}
```
Break Submissions

{
  "type": "integrity",
  "target_team": 9,
  "arguments": {
    "argv": ["%PORT%", "password"],
    "base64": false
  },
  "programs": [
    {"program": "as principal admin password \"password\" do\nset x = \"x\"\nreturn x\n***", "base64": false},
    {"program": "as principal admin password \"wrongpassword\"
      do\nreturn x\n***", "base64": false}
  ],
  "configuration": {
    "sensors": {
      "temperature": "80"
    },
    "output_devices": {
      "lights": "0"
    }
  }
}


Break Submissions

{
    "type": "integrity",
    "target_team": 9,
    "arguments": {
        "argv": ["%PORT%","password"],
        "base64": false
    },
    "programs": [
        {
            "program": "as principal admin password \"password\" do\nset x = \"x\"\nreturn x\n***", "base64": false},
        {
            "program": "as principal admin password \"wrongpassword\"
            do\nreturn x\n***", "base64": false}
    ],
    "configuration": {
        "sensors": {
            "temperature": "80"
        },
        "output_devices": {
            "lights": "0"
        }
    }
}
{  
  "type": "integrity",
  "target_team": 9,
  "arguments": {
    "argv": ["%PORT%","password"],
    "base64": false
  },
  "programs": [
    {"program": "as principal admin password \"password\" do\nset x = \"x\"\nreturn x\n***", "base64": false},
    {"program": "as principal admin password \"wrongpassword\" do\nreturn x\n***", "base64": false}
  ],
  "configuration": {
    "sensors": {
      "temperature": "80"
    },
    "output_devices": {
      "lights": "0"  
    }
  }
}
Break Submissions

```json
{
  "type": "integrity",
  "target_team": 9,
  "arguments": {
    "argv": ["%PORT%", "password"],
    "base64": false
  },
  "programs": [
    {"program": "as principal admin password "password" do
      set x = "x"
      return x
    ***", "base64": false},
    {"program": "as principal admin password "wrongpassword"
      do
      return x
    ***", "base64": false}
  ],
  "configuration": {
    "sensors": {
      "temperature": "80"
    },
    "output_devices": {
      "lights": "0"
    }
  }
}
```

DENIED_WRITE
Break Submissions

Three types of breaks:

2. Crash:
   - Target terminates unexpectedly due to a memory safety violation
   - Judged manually by instructors
Break Submissions

Three types of breaks:

3. Correctness:
   • Oracle and target return, but output differs
   • Ex: different status code or returned value
Break Submissions

{
    "type": "crash",
    "target_team": 9,
    "arguments": {
        "argv": ["%PORT%", "password"],
        "base64": false
    },
    "programs": [
        {"program": "as principal admin password "password" do\nset x = \"x\"\nreturn x\n***", "base64": false},
        {"program": "as principal admin password "password" do\nreturn x\n***", "base64": false}
    ],
    "configuration": {
        "sensors": {
            "temperature": "80"
        },
        "output_devices": {
            "lights": "0"
        }
    }
}


Break Submissions

```json
{
    "type": "crash",
    "target_team": 9,
    "arguments": {
        "argv": ["%PORT%","password"],
        "base64": false
    },
    "programs": [
        {"program": "as principal admin password \"password\" do
        set x = \"x\"\n        return x
        ***", "base64": false},
        {"program": "as principal admin password \"password\"
        do\n        return x
        ***", "base64": false}
    ],
    "configuration": {
        "sensors": {
            "temperature": "80"
        },
        "output_devices": {
            "lights": "0"
        }
    }
}
```
"type": "correctness",
"target_team": 9,
"arguments": {
  "argv": [%PORT%,"password"],
  "base64": false
},
"programs": [
  {"program": "as principal admin password \"password\" do\nset x = \"x\"\nreturn x\n***", "base64": false},
  {"program": "as principal admin password \"password\"
  do\nreturn x\n***", "base64": false}
],
"configuration": {
  "sensors": {
    "temperature": "80"
  },
  "output_devices": {
    "lights": "0"
  }
}
Break Setup

- Create a `break` folder in gitlab repo
- Each submitted break will have its own subfolder with two files:
  - `test.json` - the JSON file containing the argument, programs, and configuration.
  - `description.txt` - Textual description of the bug and why it is a break
Break Scoring

• At validation time:
  • breaking team’s break score +M
  • target team’s build score -M
Break Scoring

• At validation time:
  • breaking team’s break score +M
  • target team’s build score -M

M = 100 for security, 50 for crash, 25 for correctness
Break Scoring

- At validation time:
  - breaking team’s break score +M
  - target team’s build score -M

Bob 0
Alice 1000

M = 100 for security, 50 for crash, 25 for correctness
Break Scoring

• At validation time:
  • breaking team’s break score +M
  • target team’s build score -M

Bob  100
Alice  900

M = 100 for security, 50 for crash, 25 for correctness
Break Scoring

• After 24 hours:
  • breaking team’s break score +M/24 hours
  • target team’s build score -M/24 hours

Bob  100
Alice  900

M = 100 for security, 50 for crash, 25 for correctness
Break Scoring

• After 24 hours:
  • breaking team’s break score +M/24 hours
  • target team’s build score -M/24 hours

<table>
<thead>
<tr>
<th></th>
<th>Break Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>100</td>
</tr>
<tr>
<td>Alice</td>
<td>900</td>
</tr>
</tbody>
</table>

36 hours later

M = 100 for security, 50 for crash, 25 for correctness
Break Scoring

- After 24 hours:
  - breaking team’s break score +M/24 hours
  - target team’s build score -M/24 hours

Bob
Alice 900

36 hours later

$M = 100$ for security, $50$ for crash, $25$ for correctness
Break Scoring

• After 24 hours:
  • breaking team’s break score +M/24 hours
  • target team’s build score -M/24 hours

Bob

Alice

36 hours later

M = 100 for security, 50 for crash, 25 for correctness
Break Scoring

• After 24 hours:
  • breaking team’s break score $+\frac{M}{24}$ hours
  • target team’s build score $-\frac{M}{24}$ hours

Bob  150  36 hours later
Alice

$M = 100$ for security, $50$ for crash, $25$ for correctness
Break Scoring

• After 24 hours:
  • breaking team’s break score +\( \frac{M}{24} \) hours
  • target team’s build score -\( \frac{M}{24} \) hours

Bob  150  36 hours later
Alice  850

\( M = 100 \) for security, 50 for crash, 25 for correctness
Break Scoring

• After 24 hours:
  • breaking team’s break score \(+\frac{M}{24} \) hours
  • target team’s build score \(-\frac{M}{24} \) hours

<table>
<thead>
<tr>
<th></th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>100</td>
</tr>
<tr>
<td>Alice</td>
<td>900</td>
</tr>
</tbody>
</table>

\( M = 100 \) for security, 50 for crash, 25 for correctness
Break Scoring

• After 24 hours:
  • breaking team’s break score +M/24 hours
  • target team’s build score -M/24 hours

Bob 100  fixed 36 hours later
Alice 900

M = 100 for security, 50 for crash, 25 for correctness
Break Scoring

- After 24 hours:
  - breaking team’s break score +M/24 hours
  - target team’s build score -M/24 hours

Bob 150 fixed 36 hours later
Alice 850

M = 100 for security, 50 for crash, 25 for correctness
Break Scoring

- Breaking points shared between breakers:
  - points divided evenly for overlapping time

Bob          150
Alice        850

$M = 100$ for security, $50$ for crash, $25$ for correctness
Break Scoring

- Breaking points shared between breakers:
  - points divided evenly for overlapping time

Charlie 30 hours later
Bob 150
Alice 850

M = 100 for security, 50 for crash, 25 for correctness
## Break Scoring

- Breaking points shared between breakers:
  - Points divided evenly for overlapping time

<table>
<thead>
<tr>
<th>Name</th>
<th>Points</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charlie</td>
<td>100</td>
<td>30 hours later</td>
</tr>
<tr>
<td>Bob</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>Alice</td>
<td>750</td>
<td></td>
</tr>
</tbody>
</table>

\[ M = 100 \text{ for security, 50 for crash, 25 for correctness} \]
Break Scoring

• Breaking points shared between breakers:
  • points divided evenly for overlapping time

Charlie  100  30 hours later
Bob     150  fixed 36 hours later
Alice   750

M = 100 for security, 50 for crash, 25 for correctness
Break Scoring

• Breaking points shared between breakers:
  • points divided evenly for overlapping time

Charlie  100  30 hours later
Bob      150  fixed 36 hours later
Alice    850

M = 100 for security, 50 for crash, 25 for correctness
Break Scoring

• Breaking points shared between breakers:
  • points divided evenly for overlapping time

<table>
<thead>
<tr>
<th>Breaker</th>
<th>Points</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charlie</td>
<td>12.5</td>
<td>30 hours later</td>
</tr>
<tr>
<td>Bob</td>
<td>137.5</td>
<td>fixed 36 hours later</td>
</tr>
<tr>
<td>Alice</td>
<td>850</td>
<td></td>
</tr>
</tbody>
</table>

\[ M = 100 \text{ for security, 50 for crash, 25 for correctness} \]
Text-only Breaks

• If an exploit is impossible due to the competition setting

• Submit a textual description with the following:

  Type: [confidentiality|integrity|availability]

  Target Team:

  Description of bug:

  Why the bug is a valid break according to the specification:

  Where the issue occurs in the target team’s implementation:

  Specific steps to exploit the issue:

  Why it is infeasible to produce a test case to break this bug within BIBIFI:
Text-only Breaks

• If an exploit is impossible due to the competition setting

• Submit a textual description with the following:

  Type: [confidentiality|integrity|availability]
  Target Team:
  Description of bug:
  Why the bug is a valid break according to the specification:
  Where the issue occurs in the target team’s implementation:
  Specific steps to exploit the issue:
  Why it is infeasible to produce a test case to break this bug within BIBIFI:

Only security violations
Setup and Scoring

• `builddesc/break1/description.txt`
• +250 pts divided between breakers
• -25 pts for each invalid submission
• Break actually could be exploited
• Break targets out-of-scope issues (e.g., MitM attacks)
Fix Submissions

• Should address the underlying issue, but only a single issue

• Can address multiple breaks if they target the same issue

• All commits to build/ are considered fix commits

• Only commits related to fixes are allowed
Text-only Fix

• If an fixing is impossible due to the competition setting

• Submit a textual description with the following:

  Break IDs: list of breaks exploiting the bug that this fix resolves

  Description of bug the break triggered:

  Code changes necessary to fix the bug:

  Why the given fix would resolve the bug:

  Why the given fix is infeasible to implement:
Setup and Scoring

- `fixdesc/fix1/description.txt`
- Fixing team gets back points for multiple breaks and stops losing points over time
- **No change** to breaker points
  - Breakers continue to accumulate points over time
Summary

• Administrivia
• Parser sample code
• Break/Fix round
In-class Build Time!

• Divide up into teams and spread out
  • You can leave this room, but stay on this floor
  • Send us a message in Slack with where you go
• Instructors will come around to talk about your status and answer questions