Hour of Code: Teacher Guide

CodeHS Cyber Range | High School Cyber Missions

Ten realistic cybersecurity simulations where students hack a login page, unmask a deepfake, analyze malware, trick an AI chatbot, and investigate a hacked account. No prior coding or cybersecurity knowledge needed.

Audience	Grades 9-12. No prior experience needed.
Time	About 60 minutes (5 minute opener, 45-50 minutes of missions, 5-10 minute wrap-up)
Student link	**codehscyberrange.com/hoc**
Mission passcode	**`secure-shark-97`**
Accounts	None. Students do not sign up or enter any personal information.
Materials	One computer or Chromebook per student (or per pair) with an up-to-date web browser.

Before the Hour of Code

- **Test it on a student device.** Open **codehscyberrange.com/hoc** on the same kind of computer and network students will use, enter the passcode, and open one activity. If the page is blocked, ask your IT team to allow **codehscyberrange.com**.
- **Try a mission yourself.** Playing the first activity takes about 10 minutes and is the best way to get comfortable before class.
- **Read the Teacher Notes** at the back of this guide. Each activity has an overview, objectives, discussion questions, and a step-by-step solution.
- **Write the link and passcode on the board** so students can get started right away.

During the Hour of Code

1. **Open with a question (5 minutes).** Ask: *"What's the biggest hack or scam you've heard about in the news? How do you think it happened, and who do you think stopped it?"* Explain that today they will try the real work cybersecurity professionals do.
2. **Get students in.** Students go to **codehscyberrange.com/hoc**, type the mission passcode **secure-shark-97**, and click **Unlock your mission**. The passcode works on that browser for 24 hours.
3. **Students work at their own pace.** They open activities from the mission pathway in order. Each activity has a short story and directions, and a guide character walks them through every step.

4. **Capture the flag.** At the end of each activity, students unlock a career card with a flag that looks like **flag{...}**. They copy it, paste it into the **Found the flag?** box, and submit to clear the activity. Every student's flags are unique, so flags can't be shared.
5. **Keep everyone moving.** Most students complete four to six activities in an hour. There are ten so that every student always has a next mission, and the set works well across two sessions.
6. **Wrap up (5-10 minutes).** Bring the class together with the closing discussion questions below.

Closing Discussion Questions

- Which activity felt most like a job you could see yourself doing? Why?
- Several activities showed attackers tricking people or AI rather than breaking code. What does that tell you about security?
- In which activities did you act as an attacker? Why do companies hire people to attack them?
- What is one habit you will change after today to protect your own accounts?

Hour of Code Tips

- **It's okay not to know the answers.** Every activity is designed for students with no background. Learning alongside your students is part of the fun.
- **Point students back to the guide.** When a student is stuck, ask them to reread the guide's last message and the directions before you step in. Wrong answers come with an explanation, not a penalty.
- **Use prompting questions** instead of giving answers: *What is the activity asking you to find? What clue have you not used yet? What would an attacker look for here?*
- **Progress is saved on the browser.** Students should use the same computer and browser for the whole session. Switching devices or clearing the browser starts them over.
- **Shared computers:** when a student is completely done, they can click **End mission & clear progress** so the next class starts fresh. Remind students not to click it mid-session.
- **Pair up if needed.** These activities work well with two students per device. Have partners switch who drives after each activity.
- **Everything is a safe simulation.** Students attack only fictional systems, and every activity reminds them that real hacking always requires permission.

Closing

Thank you for bringing cybersecurity to your students! We hope this hour sparks their curiosity about the many careers in cyber. If you have questions or feedback, reach out to us at **hello@codehs.com**.

Activity Overview

Activities appear on the mission pathway in this order. Times are estimates for a student working alone.

#	Activity	Topic	Career unlocked	Time
1	Injection Point (Hour of Code)	Web Security	Application Security Engineer	10-12 min
2	Deepfake Desk (Hour of Code)	Digital Forensics	Threat Intelligence Analyst	10-12 min
3	Ghost Trail (Hour of Code)	Digital Forensics	OSINT Investigator	12-15 min
4	Smart Home Takeover (Hour of Code)	Network Security	IoT Security Engineer	10-12 min
5	Cipher Breaker (Hour of Code)	Cryptography	Cryptanalyst	12-15 min
6	Log Hunter (Hour of Code)	Linux	Security Operations (SOC) Analyst	12-15 min
7	Malware Lab (Hour of Code)	Digital Forensics	Malware Analyst	12-15 min
8	Prompt Injection (Hour of Code)	Web Security	AI Security Engineer	10-12 min
9	Red vs. Blue (Hour of Code)	Network Security	Red Team Operator	12-15 min
10	Impossible Travel (Hour of Code)	Digital Forensics	Identity Security Analyst	12-15 min

Teacher Notes: Activity 1

Injection Point (Hour of Code)

Web Security | About 10-12 minutes | Career unlocked: **Application Security Engineer**

Overview

Students test a school login page before launch. They watch the SQL query build live as they type, break it with a single quote, log in as admin without a password using SQL injection, then apply a fix and re-test.

Objectives

Students will be able to:

- Explain how a website turns login input into a database query
- Describe how SQL injection changes what a query does
- Explain why parameterized queries prevent SQL injection

Discussion Questions

- Why did one quote mark break the whole query?
- Where else do websites take input that could be misused this way?
- Why is re-testing after a fix an important step?

Solution

1. Log in with username jdoe and password pancakes42 to see a query that matches exactly 1 row.
2. Log in as jdoe with any wrong password to see 0 rows and a failed login.
3. Type a single quote (') in the username and log in to cause a SQL syntax error.
4. Enter admin'-- as the username with any password. The -- turns the rest of the query, including the password check, into a comment, so you are logged in as admin. (' OR '1'='1' -- also works.)
5. Choose the parameterized query patch, then try admin'-- again and confirm the login is blocked.
6. Copy the unique flag from the career card, paste it into the Found the flag? box, and submit.

Why it matters

Students see exactly how user input becomes part of a database query, discover that a single quote can break the query's structure, and use comments to bypass a password check. Patching with parameterized queries and re-testing teaches the core secure coding rule: never mix untrusted input with code.

Teacher Notes: Activity 2

Deepfake Desk (Hour of Code)

Digital Forensics | About 10-12 minutes | Career unlocked: **Threat Intelligence Analyst**

Overview

As a finance intern, students get a voicemail and video call from what appears to be the CEO asking for an urgent $48,000 wire. They find red flags, sort real deepfake clues from false ones, verify through a trusted channel, and pick company policies that stop this scam.

Objectives

Students will be able to:

- Define deepfake and social engineering
- Identify warning signs of AI-powered impersonation
- Explain out-of-band verification and why process beats spotting fakes

Discussion Questions

- If a voice and face can be faked, how can you know who you're talking to?
- What would you do if a family member called asking for money urgently?
- Why do companies use policies like two-person approval?

Solution

1. In the voicemail, click the four red flags: boarding a flight so she can't be reached, wiring $48,000 to a new account, the 30-minute deadline, and the request to keep it secret from the finance team.
2. Mark lip-sync lag, never turning her head, shimmering hair edges, and freezing when asked to wave as deepfake clues. Mark her voice sounding right and the company hoodie as not clues.
3. Choose to look up Dana's number in the company directory and call her directly, then call Dana Whitfield's real number to confirm she never made the request.
4. Pick two-person approval for large payments, call-back verification using a known number, and a secret code phrase, then submit.
5. Copy the unique flag from the career card, paste it into the Found the flag? box, and submit.

Why it matters

Students practice recognizing CEO fraud and AI-powered impersonation, learn that a convincing voice or face is no longer proof of identity, and apply out-of-band verification. Choosing process-based defenses shows why organizations rely on procedures that work even against a perfect deepfake.

Teacher Notes: Activity 3

Ghost Trail (Hour of Code)

Digital Forensics | About 12-15 minutes | Career unlocked: **OSINT Investigator**

Overview

Students help police identify a scam seller by pivoting from a username to an old email, a gamer tag, and a real name, then geolocate a photo using street signs and landmarks. They finish by handling the evidence ethically and choosing habits that protect their own privacy.

Objectives

Students will be able to:

- Define OSINT and pivoting
- Explain how reused usernames and emails link accounts together
- Describe the ethics of handling information found online

Discussion Questions

- If someone searched your usernames, what could they connect?
- Why is it wrong to post the seller's name publicly, even if they're guilty?
- What one change will you make to your own online habits?

Solution

1. Search kicks4less_drop to find an old ForumHub post listing the email k4l.deals@mailbox.example.
2. Search k4l.deals@mailbox.example to find a GameZone profile with the gamer tag RVoss_88.
3. Search RVoss_88 to find a PicPost profile with the real name Ryan Voss.
4. In the photo, click the HARBOR ST sign, the Route 42 Lakeside bus, The Gull & Kettle cafe, and the lighthouse, then click View photo details.
5. On the map, choose Lakeside, the only neighborhood with Harbor St, bus Route 42, and The Gull & Kettle.
6. Choose to send the evidence to the police and the marketplace safety team, then pick different usernames, careful photos, and a separate public email as protective habits.
7. Copy the unique flag from the career card, paste it into the Found the flag? box, and submit.

Why it matters

Students experience how a single reused email or username connects separate accounts, and how everyday photos reveal locations even with GPS data removed. Choosing to report instead of doxxing or confronting a suspect, then applying the lesson to their own habits, emphasizes the ethics and personal privacy side of OSINT.

Teacher Notes: Activity 4

Smart Home Takeover (Hour of Code)

Network Security | About 10-12 minutes | Career unlocked: **IoT Security Engineer**

Overview

With the family's permission, students scan a smart home network, find the riskiest device from its open ports, take over two devices with published default passwords, apply fixes, and re-test to prove the house is secure.

Objectives

Students will be able to:

- Define IoT, IP address, and port
- Explain why default passwords are dangerous
- Follow the scan, exploit, fix, and verify cycle of a penetration test

Discussion Questions

- What smart devices are in your home or school? Do you know their passwords?
- Why do manufacturers ship devices with default passwords?
- Why does putting smart devices on their own network help?

Solution

1. Click Scan network, then click the Living Room Camera, which has Telnet (port 23) open along with a video stream.
2. Log in to the Living Room Camera with admin / 1234 and the Thermostat with admin / admin from the public manual list. The router and smart lock do not use default passwords.
3. Apply the six real fixes: change the camera password, update the camera firmware, turn off Telnet, change the thermostat password, update the thermostat firmware, and move smart devices to a separate IoT network. Use passwords of at least 12 characters, and skip the two traps.
4. Re-test the camera with admin / 1234 to confirm access is denied.
5. Copy the unique flag from the career card, paste it into the Found the flag? box, and submit.

Why it matters

Students follow a full penetration test cycle (scan, exploit, fix, verify) against smart home devices, learning what IP addresses and ports reveal and why default passwords are so dangerous. Re-testing after hardening reinforces that security work isn't done until the fix is proven.

Teacher Notes: Activity 5

Cipher Breaker (Hour of Code)

Cryptography | About 12-15 minutes | Career unlocked: **Cryptanalyst**

Overview

Students crack an intercepted substitution cipher using frequency analysis and word patterns, the same methods codebreakers used in World War II. The key is randomized for each student.

Objectives

Students will be able to:

- Explain how a substitution cipher works
- Use frequency analysis and word patterns to break a cipher
- Explain why modern encryption is designed to hide patterns

Discussion Questions

- Why did knowing E is the most common letter help so much?
- Would this method work on a very short message? Why or why not?
- How is modern encryption different from this cipher?

Solution

1. Click the tallest bar in the frequency chart (the most common code letter) and map it to E.
2. Find the 3-letter word that appears many times and ends in E, and map its first two letters to T and H.
3. Map the one-letter word to A, then crack THIS (I and S) and CODE (C, O, D) from the clues.
4. Fill in the remaining letters using word patterns such as THAT, WRONG, LETTER, and PASSWORD, using Check my work or Reveal a letter as needed.
5. Read the decoded message and enter the password LIGHTHOUSE.
6. Copy the unique flag from the career card, paste it into the Found the flag? box, and submit.

Why it matters

Students break a substitution cipher using frequency analysis and word patterns, the same techniques historical codebreakers used. Because the key is randomized for every student, they must reason through the patterns rather than copy an answer, and the ending explains why modern encryption is designed to leave no patterns behind.

Teacher Notes: Activity 6

Log Hunter (Hour of Code)

Linux | About 12-15 minutes | Career unlocked: **Security Operations (SOC) Analyst**

Overview

In a simulated Linux terminal, students use ls, cd, cat, grep, and a pipe to search a 5,000-line login log, count failed logins for three suspect IPs, discover the attacker got in, and block them with a firewall command.

Objectives

Students will be able to:

- Use basic Linux commands to read and search files
- Use a pipe to send one command's output into another
- Use counts to tell a brute force attack apart from normal mistakes

Discussion Questions

- Why couldn't you just read the log file yourself?
- Why did the IP with 3 failed logins probably not belong to an attacker?
- What would you do next after blocking the attacker?

Solution

1. Run ls, then cat README.txt to read the ticket listing three suspect IPs: 203.0.113.66, 198.51.100.23, and 192.0.2.44.
2. Run cd logs, then cat auth.log to see the huge log file.
3. Run grep FAILED auth.log to show only failed logins, then grep FAILED auth.log | wc -l to count them.
4. Count the lines for each suspect with grep IP auth.log | wc -l. 203.0.113.66 has 1,477, 198.51.100.23 has 63, and 192.0.2.44 has 3. Click 203.0.113.66 in the evidence table as the attacker.
5. Run grep 203.0.113.66 auth.log | grep SUCCESS to discover the attacker logged in as admin.
6. Run sudo ufw deny from 203.0.113.66 to block the attacker.
7. Copy the unique flag from the career card, paste it into the Found the flag? box, and submit.

Why it matters

Students learn a small set of beginner Linux commands (ls, cd, cat, grep) and the idea of a pipe, then use them to turn thousands of log lines into evidence. Comparing line counts for three IPs separates a brute force attack from a password spray and a normal typo, and the investigation ends with a real incident response action.

Teacher Notes: Activity 7

Malware Lab (Hour of Code)

Digital Forensics | About 12-15 minutes | Career unlocked: **Malware Analyst**

Overview

Students investigate a fake 'Chapter 5 notes' file in a sandbox. They unmask its real file type, check its hash, find suspicious strings, run it to watch its behavior, and write a detection rule that catches it without flagging clean files.

Objectives

Students will be able to:

- Explain the difference between static and dynamic analysis
- Explain why a hash lookup can miss new malware
- Write a detection rule and test it for false positives

Discussion Questions

- Why did 0 of 62 scanners flagging the file not prove it was safe?
- Why is it important to run malware only in a sandbox?
- What's the risk of a detection rule that's too broad?

Solution

1. Compare the first bytes 4D 5A (MZ) to the cheat sheet and choose that it's a Windows program pretending to be a PDF.
2. Calculate the hash, search ThreatVault, and answer that 0 of 62 doesn't mean safe because the file may be brand-new malware.
3. Click the four suspicious strings: the 198.51.100.77/upload address, the AppData svchost.exe path, the CurrentVersion Run registry key, and the Chrome Login Data path.
4. Detonate the file and label the behaviors: decoy PDF and svchost copy as Hiding, Run key as Staying, reading Login Data as Stealing, and sending data to 198.51.100.77 as Phoning home.
5. Build a rule with strings only the malware has, such as 198.51.100.77/upload and the AppData svchost.exe path, then test it to catch both samples with zero false positives.
6. Copy the unique flag from the career card, paste it into the Found the flag? box, and submit.

Why it matters

Students walk through the malware analysis workflow: unmasking a disguised file type, learning why hash lookups miss new malware, extracting strings, observing behavior in a sandbox, and writing a YARA-style detection rule. Testing the rule against clean files teaches the real-world trade-off between catching threats and avoiding false positives.

Teacher Notes: Activity 8

Prompt Injection (Hour of Code)

Web Security | About 10-12 minutes | Career unlocked: **AI Security Engineer**

Overview

As an AI red team, students try to trick a school chatbot into revealing a secret exam code hidden in its instructions. They leak it two different ways, then choose guardrails that work even if the AI can be tricked.

Objectives

Students will be able to:

- Define system prompt and prompt injection
- Explain why an AI model can't reliably keep secrets in its instructions
- Choose secure design defenses such as removing secrets and least privilege

Discussion Questions

- Why couldn't RidgeBot just follow its rule to never share the code?
- What kinds of information should never be put in a chatbot's instructions?
- Where do you see AI chatbots used? What could go wrong?

Solution

1. Ask a normal question, such as what's for lunch, and get an answer.
2. Ask directly for the exam code and see RidgeBot refuse.
3. Use two different techniques to leak BLUEJAY-7, for example asking the bot to repeat its instructions word for word, and claiming to be a teacher. Asking it to ignore its previous instructions, role-play a story, or spell the answer letter by letter also works.
4. Choose the three strong guardrails: remove the code from the bot's instructions, filter replies for the secret, and limit the bot to only the information it needs.
5. Try an attack again on the patched bot and confirm it has nothing to leak.
6. Copy the unique flag from the career card, paste it into the Found the flag? box, and submit.

Why it matters

Students experience why AI systems can't reliably keep secrets placed in their instructions, try several real categories of prompt injection, and learn secure design principles (removing secrets, output filtering, and least privilege) that work even when the model can be tricked. The simulation is simplified, but the attack types and defenses are real.

Teacher Notes: Activity 9

Red vs. Blue (Hour of Code)

Network Security | About 12-15 minutes | Career unlocked: **Red Team Operator**

Overview

Students first play the red team, choosing a working move for each of five attack steps against a company. Then they switch to the blue team and spend a $100k budget on defenses that stop that exact attack.

Objectives

Students will be able to:

- Describe the steps of a typical attack path
- Connect each attack step to the weakness that made it possible
- Explain defense in depth and make budget-based security decisions

Discussion Questions

- Which attack step surprised you most? Why?
- Why is one strong defense not enough?
- How did the budget change your choices?

Solution

1. Scout: research employees on social media to pick IT admin Jamie Ortiz as the target.
2. Get in: email Jamie a shipping invoice document with a hidden macro, since the email filter lets macro documents through.
3. Get admin: secretly record Jamie's keystrokes to capture the admin password.
4. Spread: try Jamie's admin password on every server, since admins reuse one password on a flat network.
5. Steal: upload the data in small pieces overnight, since nobody watches outgoing data.
6. As the blue team, choose defenses that cover all 5 steps within $100k, such as social media training ($10k), blocking macros ($15k), MFA for admin logins ($25k), and monitoring data leaving the network ($20k), then deploy and replay the attack.
7. Copy the unique flag from the career card, paste it into the Found the flag? box, and submit.

Why it matters

Students see a realistic multi-step attack from the attacker's perspective, connect each step to a specific weakness, then make budget-constrained defensive decisions. Avoiding an expensive control that doesn't address the actual attack, and layering defenses across every step, teaches risk-based thinking and defense in depth.

Teacher Notes: Activity 10

Impossible Travel (Hour of Code)

Digital Forensics | About 12-15 minutes | Career unlocked: **Identity Security Analyst**

Overview

A school staff account was taken over and used to send a scam to 412 parents. Students find two sign-ins that are physically impossible, figure out how the attacker beat MFA, flag the attacker's actions, and build a full cleanup plan.

Objectives

Students will be able to:

- Explain impossible travel and how sign-in logs reveal it
- Describe an MFA fatigue attack
- Plan the steps to recover a compromised account

Discussion Questions

- Why did MFA not stop this attacker?
- What should you do if you get an MFA prompt you didn't expect?
- Why did the attacker create a hidden inbox rule?

Solution

1. Compare the Tuesday 7:35 AM sign-in from Denver with the 7:52 AM sign-in from Sydney. They are about 8,300 miles apart in 17 minutes, which is impossible travel.
2. Choose that the attacker spammed Ms. Alvarez's phone with approval pop-ups until she tapped Approve (an MFA fatigue attack).
3. On the phone, choose Deny and report: This wasn't me.
4. In the activity log, flag the four actions from the Linux PC in Sydney: the hidden inbox rule, downloading the parent contact list, sending 412 scam emails, and changing the recovery phone number.
5. Choose the five cleanup actions: sign out all sessions, reset the password, remove the inbox rule and fix the recovery phone, switch MFA to number matching, and warn parents the email was fake.
6. Copy the unique flag from the career card, paste it into the Found the flag? box, and submit.

Why it matters

Students investigate an account takeover using sign-in logs, calculate why two logins are physically impossible, and recognize MFA fatigue, a real technique attackers use to get around push-based MFA. Separating attacker actions from normal activity by location and device, then building a full cleanup plan, mirrors how identity security teams respond.