The CipherFiles

STORY BY

BY: MR.KENDELL

Noah Anderson had always been good with computers. He was one of the top students in Mr. Kendell's Creative Coding class, where he learned Python programming and how to troubleshoot code. While other students just followed assignments, Noah went further. He rewrote scripts to make them better, figured out how different loops and conditionals worked together, and often helped fix his classmates' broken code. When something didn't work, he had to know why.

Noah also had a thing for nicknames. He loved coming up with names for his friends, especially ones that sounded cool or tech-related. He and his group started calling Mr. Kendell "Master K," and before long, the whole school used it. But Noah himself didn't have a nickname, and even though he wanted one, nothing ever stuck. He figured he'd earn one eventually—something that truly fit.

Across the school, Damian Dalton was working on something different. While Noah spent his time fixing code, Damian was busy breaking it. He liked finding weaknesses in the school's systems. At first, it was just a game—messing with the bell schedule, causing random login errors, or making small vending machine malfunctions. But the more he experimented, the more he realized how much control he could take.

Damian didn't just want to cause problems—he wanted to prove something. He believed that technology ruled everything, and if he could control the system, he could control the school. What he didn't know was that every time he messed with something, Noah was the one fixing it. While Damian thought of Noah as a rival, Noah didn't even know he was in a battle.

Noah didn't see himself as a hacker. He loved solving problems, not creating them. When a program had an error, he fixed it. When a computer froze, he figured out why. It never crossed his mind that someone could be causing these problems on purpose. So when the library's self-checkout system started failing, Noah just assumed it was a software bug. Books weren't scanning, student accounts weren't updating, and the librarian was frustrated. Noah took a look, found the issue, and patched it. He never thought to ask how the error got there in the first place.

Meanwhile, Damian was watching and learning. He knew someone was fixing the things he was breaking, but he didn't know who—yet. Every time he caused a glitch, it disappeared within a day or two. That meant someone was cleaning up after him. At first, it annoyed him. But then, he saw it as a challenge. If someone was going to erase his work, he would make sure it wasn't so easy next time.

The next few "glitches" became more subtle. The school's vending machines started rejecting student ID payments, random error messages popped up on login screens, and some classroom projectors refused to connect to teacher laptops. None of these were major problems, but they were annoying enough to cause distractions. Students complained, teachers grumbled, and Noah—thinking it was just another day of school tech issues—kept stepping in to fix them.

But as the tech issues piled up, Noah started to notice something strange. These weren't random malfunctions—they were too specific, too deliberate. The vending machines worked fine after school, but not during lunch rush. The login errors happened right before major assignments were due. The projectors failed only when teachers needed them most. It was almost like... someone was doing this on purpose.

Noah wasn't supposed to have access to the school's systems, but being an office aide came with perks. The principal knew he was tech-savvy, and whenever there was a computer issue too big for the teachers to handle, Noah was often the one called in to help. Sometimes, the principal even pulled him out of class to fix urgent problems—anything from locked accounts to network crashes. Because of this, Noah was given temporary admin access at times, allowing him to see how the school's systems worked on a deeper level.

Damian, however, had no such permission—but that didn't stop him. He wasn't fixing things; he was breaking them on purpose. Unlike Noah, who learned through official coding lessons, Damian had taught himself through trial and error. He had discovered security flaws in the school's outdated network, finding ways to bypass certain restrictions. While most students were locked out of protected files, Damian had figured out how to slip through unnoticed. Every system had a weakness, and Damian was determined to find them all.

The first time Noah noticed something was off, he was fixing a problem with the library's checkout system. It should have been a simple bug—just a missing line of code. But when he checked the system logs, he found extra commands that shouldn't have been there. It was like someone had intentionally messed with the code. The problem wasn't just an error—someone had changed it on purpose.

Meanwhile, Damian was getting bolder. He had started with small glitches, but now he wanted to see how much control he really had. He tested his limits by turning off certain Wi-Fi access points, resetting passwords, and making school printers spit out pages of gibberish. Nobody suspected a student could be behind it—except Noah, who was starting to put the pieces together.

At first, Noah didn't think much of the strange code changes he had found. The school's tech wasn't perfect, and bugs happened all the time. But as he kept working on different school systems, he started to notice a pattern. The errors weren't random—they were deliberate. Each time he fixed a problem, it was as if someone had intentionally altered the code, then left it to see what would happen. It didn't feel like a mistake. It felt like a test.

Damian, meanwhile, had moved past small tricks. He wasn't just causing glitches—he was watching how people reacted. Each time Noah unknowingly fixed his work, Damian studied the logs and learned how the school reset its systems. This gave him even more ideas for future exploits. He started messing with classroom projectors, making them flicker or refuse to connect. He edited digital announcements, inserting tiny errors that made messages unreadable. These weren't just pranks—they were experiments.

Noah's suspicions grew when the same types of errors kept appearing. The coding issues weren't scattered—they were targeted. It was always something noticeable enough to cause frustration but not serious enough to raise alarms. It was as if someone wanted to see how far they could push before getting caught. The realization made him uneasy. Someone wasn't just messing with the school's tech—they were learning how to control it. But who? No students should have had access to these systems. The only other people with the right permissions were teachers and IT staff. Noah couldn't imagine a teacher wasting time on this. That left only one possibility—someone had found a way in without permission. And whoever they were, they were getting better.

INTRODUCTION RECAP

Chapter 1: The Bell Schedule Hack

Paragraph 1: A System Out of Sync (Syntax Error)

Noah Anderson didn't mind fixing code. In fact, he enjoyed it. Whether it was a broken program or a miswritten command, he saw every error as a puzzle to solve. But when the school's bell schedule malfunctioned, it wasn't a simple fix—it was chaos. Bells rang at the wrong times, some didn't ring at all, and students wandered the halls confused. Teachers blamed a system glitch, but Noah suspected something deeper. He accessed the scheduling code and immediately spotted a syntax error—a misplaced colon in the timing function. It was a small mistake, but one that could have thrown the entire schedule off balance. With a few keystrokes, he corrected it, assuming it was just a careless programming mistake. What he didn't know was that someone had placed it there on purpose.

Paragraph 2: A Timed Disruption (Integer)

Across the school, Damian Dalton smirked as he watched the confusion unfold. His plan had been simple—manipulate the bell system's timing by adjusting a few key values. Instead of using the usual integer values for minutes, he had altered them just enough to throw everything off. An integer, a whole number like 10 or 15, dictated when the bells should ring. By shifting these values slightly, he caused classes to start and end at unpredictable moments. But his favorite change? He made lunch last twenty extra minutes by increasing the integer value for that period. Students celebrated, teachers were frustrated, and administrators scrambled for answers. It wasn't enough to break the system entirely—just enough to make people second-guess the schedule. When he saw Noah head toward the office, Damian knew the game had begun.

Paragraph 3: Tracing the Problem (Float)

Noah sat at the front office computer, scanning through the bell schedule's logs. Something didn't add up. The system had specific integer values for when the bells should ring, but mixed in with them were unexpected float values—numbers with decimal points. Instead of ringing exactly on time, some bells were set to 11.47 or 2.03, causing small but noticeable delays. A float was typically used for more precise calculations, not something as straightforward as a school schedule. Noah frowned. This wasn't just a simple syntax error—someone had deliberately tampered with the settings. But why? And more importantly, how?

Paragraph 4: The Cover-Up (Argument)

Noah adjusted his glasses and began restoring the correct bell times. As he worked, he noticed that some of the changes weren't random—whoever had altered the system had used specific arguments to modify the code controlling the bells. An argument is a value passed into a command to control how it behaves, like telling a robot how many steps to take. In this case, someone had entered new arguments that changed the timing of certain periods. Lunch had been extended by exactly 20 minutes, and some class transitions were shortened to 90 seconds instead of the usual five minutes. Noah quickly replaced the altered arguments with the correct ones, unaware that someone was watching his every move, waiting to see how long it would take him to fix the mess.

Paragraph 5: Glitch's Takeaway (Logic Error)

Damian leaned back in his chair, watching the school day unfold exactly as he had planned. The bell schedule had caused just enough confusion to make students question the system, yet not enough to raise major alarms. But as he reviewed the system logs from his hidden access point, he noticed something unexpected—Noah had fixed everything faster than he anticipated. That wasn't supposed to happen. Damian had introduced what should have looked like a natural logic error—an issue where the code runs but produces unexpected results. It was like writing a program to calculate a student's average grade but mistakenly dividing by the wrong number, leading to an incorrect final score. By adjusting the bell times slightly and making lunch longer, the system technically still functioned, but in a way that disrupted the schedule. Yet Noah had identified the problem too quickly. That meant one thing: someone was watching just as closely as he was.

Paragraph 6: A Hidden Signature (Dot Notation)

Noah wasn't satisfied with just fixing the problem—he wanted to know why it had happened. As he scrolled through the bell schedule's settings, something unusual caught his eye. One of the modified values had been updated using dot notation, a method programmers use to access specific properties of an object in a program. Instead of a straightforward time entry, the system showed an override like schedule.lunchTime = 12.45. This wasn't how the school normally programmed its schedule. Someone had directly accessed the system's attributes, adjusting them manually instead of using the standard input forms. That was when it clicked—this wasn't just a glitch. Someone had deliberately changed the code.

Paragraph 7: A Pattern Emerges (Indentation)

Noah leaned in closer, scanning the modified code. Something else felt off—the indentation wasn't standard. In programming, indentation helps organize code into clear blocks, ensuring that functions and loops execute correctly. Every school system update followed a strict format, with neatly aligned lines of code. But here, the spacing was inconsistent—some lines were indented too much, others too little. It reminded Noah of beginner programmers who didn't yet understand how indentation controlled a program's flow. Whoever had done this wasn't just altering values—they were manually coding inside the system. That meant this wasn't an accident. Someone had hacked in.

Paragraph 8: The First Signature (Final Reveal for Lesson 1)

Noah was about to close the system when something unusual caught his eye—a stray line of code buried deep in the scheduling program. It didn't affect the bell times or cause an error, but it stood out because it served no real purpose. Curious, he clicked on it. The line read:

Glitch was here

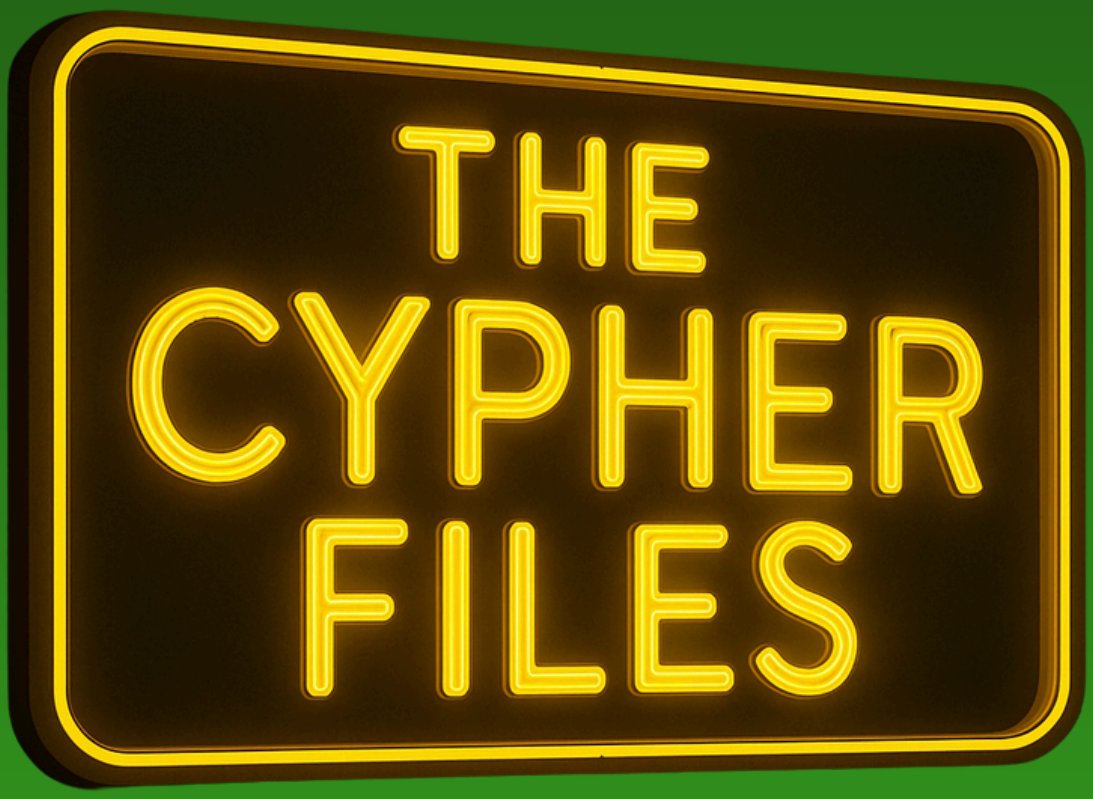
Noah's stomach tightened. This wasn't just a mistake or a system bug—someone had left a signature. Whoever had done this wanted to be noticed. His mind raced through possibilities. No student should have been able to access the bell schedule, and even if someone had tried, they wouldn't have thought to leave a message in the code. This was intentional.

He exhaled slowly, his fingers hovering over the keyboard. Was this some kind of joke? A harmless prank? Or was someone testing the system—seeing what they could change without being caught? If they could alter the bell schedule, what else could they do? Noah didn't know who was behind it yet, but one thing was certain: this was just the beginning.

At the bottom of the screen, the message remained.

Glitch was here

CHAPTER 1
RECAP



Chapter 2:

Smart TV and Projector Take over

Paragraph 1: A Normal Day... Until It's Not (Variable Name)

Noah was in the middle of coding exercises when the classroom's smartboard flickered. The teacher had just opened a lesson file, but instead of displaying the slides, the screen played a random video. At first, everyone thought it was a mistake—until students in other classrooms started reporting the same issue. Across the school, projectors and TVs turned on by themselves, playing static, outdated announcements, and even meme videos. Confusion turned into laughter in some classrooms, while in others, teachers scrambled to shut the systems down. Noah's first thought? Someone must have messed up a variable name.

A variable name is the unique identifier used in programming to store and reference data. If a programmer accidentally reuses or misspells a variable name, the program can pull the wrong information. For example, if a system was supposed to display lessonSlides but mistakenly referenced lessonVideos, it could explain why the wrong files were playing. But as Noah observed the growing chaos, he had a feeling this was no simple naming mistake—this was intentional.

Paragraph 2: Widespread Chaos (Variable)

Noah barely had time to process what was happening before more reports flooded in. Every classroom seemed to be affected. Some screens played old school announcements on repeat, while others flashed random images and glitched-out text. A few teachers managed to turn their projectors off, only for them to turn back on moments later. The principal's voice crackled through the intercom, calling for IT support immediately.

Noah frowned. This wasn't just a system error—this was deliberate. His first thought was that a variable had been misused. In programming, a variable is a container that stores a value, like a name, number, or string of text, that can change throughout a program. If a variable controlling what appears on the school's screens had been altered, it could explain why projectors and TVs were pulling up incorrect content.

But there was one problem. A mistake like that wouldn't happen everywhere at once. If this was a single variable error, it would only affect certain classrooms. Instead, the entire school's system was being manipulated at the same time. Someone had full control.

THE CYPHER FILES 2

Paragraph 3: A Mysterious Image Appears (Image Label)

As Noah scanned the room, he noticed something strange—the smartboard in the back of the classroom wasn't playing a video like the others. Instead, it displayed a distorted, glitched-out image, like a corrupted file trying to load. At first, students laughed, thinking it was a prank, but Noah felt uneasy. He had seen something like this before—when a program calls the wrong image file.

In coding, an image label is a name assigned to an image in a program so the system knows which one to display. If the wrong label is used, a different image—or no image at all—will appear instead. That's what seemed to be happening here. Instead of pulling up normal slides or videos, the system was calling corrupted or random images stored in its database.

Noah glanced at his teacher, who was frantically pressing buttons on the remote, trying to regain control. It wasn't working. That meant whoever had done this had overridden the commands entirely. Someone wasn't just messing with variables anymore—they were manipulating the system's display settings directly.

Paragraph 4: Looking for the Source (Dot Notation)

Noah pulled out his laptop and started scanning the school's network for any unusual activity. If this was just a technical glitch, the system logs would show a clear error. But deep down, he already knew—this wasn't a mistake. Someone had deliberately changed how the school's screens were pulling data.

One clue stood out. The way the screens were displaying content reminded him of dot notation. In programming, dot notation is used to access specific properties of an object, like an image, a function, or a stored file. For example, a properly written command like screen.display = lessonSlides would tell the system to show a presentation. But if someone overwrote that command with dot notation, it could be redirected to pull something else entirely—like meme videos, corrupted images, or old announcements.

Noah's suspicion deepened. Whoever had done this knew exactly what they were doing. They weren't just playing around—they were testing how much control they could take.

THE CYPHER FILES 2

Paragraph 5: The Unexpected Messages (Argument)

Noah barely had time to process what was happening before the classroom phone rang. His teacher picked it up, listened for a moment, then turned to him. "Noah, they need you in the office—now."

Noah grabbed his laptop and hurried down the hallway, passing classrooms where teachers were still struggling to turn off their projectors. When he arrived at the main office, the principal stood near the front desk, looking frustrated.

"The IT team isn't responding fast enough. Can you take a look?" the principal asked, motioning Noah toward a computer.

Noah sat down and opened the school's AV control panel. What he saw made his stomach tighten—the system logs contained messages that weren't supposed to be there.

Most of the commands looked normal, but some lines had extra arguments attached to them. In programming, an argument is a value given to a function to change how it behaves. Arguments control what gets displayed, when it appears, and even where files are pulled from. But these weren't standard commands—someone had inserted arguments that forced every screen to play unauthorized content.

Even worse, buried in the logs were eerie, out-of-place phrases like:

display.message = "Are you watching, Noah?"

Noah froze.

Whoever was doing this knew someone was fixing their work. And now, they were talking to him directly.

THE CYPHER FILES 2

Paragraph 6: Another Signature Left Behind (String Variable)

Noah’s fingers hovered over the keyboard as his mind raced. "Are you watching, Noah?" That wasn’t just a random system error. It was a direct message—someone knew he was here.

Digging deeper, Noah found a string variable buried in the code. In programming, a string variable stores text—anything from usernames to hidden messages. When he revealed the contents, a simple phrase appeared:

prankMessage = "Enjoy the show."

Noah clenched his jaw. This wasn’t a glitch. Someone had planted this on purpose.

Paragraph 7: The Bigger Picture (String Concatenation)

By using string concatenation, Noah linked multiple system logs together, revealing a timeline of attacks. The first system changes had started weeks ago, long before the bell schedule hack. This wasn’t random—it was a plan.

Paragraph 8: The Calling Card (Indentation)

The last line of code stood out. The indentation was messy—some lines too far, others misaligned. Almost like... it was meant to be noticed.

And then, at the bottom of the file, Noah found it:

Glitch was here

Noah exhaled. Who was Glitch? And how could he catch them?

Paragraph 9: Setting the Trap

Instead of just fixing the system, Noah devised a plan. He would inject his own hidden code—a silent tracker that would gather more details on Glitch without them knowing. If Glitch wanted to play, Noah would be ready.

This final version includes Noah’s plan to trap Glitch while keeping the suspense high. Let me know if this works or if you need any tweaks!

CHAPTER 2
RECAP

Chapter :3

Teacher Computer hIJACK

THE CYPHER FILES 3

Paragraph 1: The First Teacher Lockout (Comment)

Noah was halfway through his morning class when he heard frustrated voices from the hallway. At first, he ignored it—until his own teacher stopped mid-lesson and frowned at their computer.“That’s weird… I just got logged out,” they muttered, clicking furiously.A few seconds later, another teacher appeared at the door, looking just as confused. “Is your login working? Mine’s rejecting my password.”Noah’s instincts kicked in. This wasn’t normal. As the period went on, reports piled in from different classrooms—more and more teachers were getting locked out of their computers. IT was scrambling to reset passwords, but the issue wasn’t going away. Something—or someone—was causing a widespread system failure. After class, Noah made his way to the computer lab, where IT had started troubleshooting. He peeked over a technician’s shoulder as they scrolled through the login code. That’s when he saw it—strange comments buried in the code.

Just making things interesting ;)

Hope they like the inconvenience

Noah’s stomach tightened. Someone had done this on purpose.

Paragraph 2: The System Failure Spreads (Argument)

By second period, the problem had escalated. Teachers weren’t just locked out anymore—their files were vanishing or getting swapped with random documents. Some lesson plans were replaced with old test answers, while others had student report cards duplicated across multiple accounts. Then, the first major red flag appeared. A teacher in the front office yelled out in panic: “Why is my credit card information on the screen!?”Noah’s head snapped up. What?! Within minutes, other teachers reported the same terrifying issue—some of their computers were randomly pulling up sensitive documents, revealing stored credit card information, home addresses, and personal tax forms. Students who happened to be nearby took out their phones, snapping pictures before teachers could react. Noah’s pulse pounded. This wasn’t just a prank anymore—this was real damage. He pulled up the affected code and immediately spotted something suspicious: extra arguments had been added to certain commands. In programming, an argument is a value that modifies how code behaves. Normally, the system’s file retrieval process should look something like this:

getFile(userID, “lessonPlan”)
But in the corrupted code, the arguments had been altered, forcing the computers to display personal files instead.
getFile(userID, “privateData”)
Whoever had done this wasn’t just testing limits anymore—they were pushing boundaries, seeing just how much damage they could cause.
Paragraph 3: A Bizarre Event in the System (Event)

“Dalton, you totally failed Keene’s class.” The words rang out across the room, louder than Damian expected. He had barely been paying attention when a classmate next to him laughed and nudged his shoulder, saying it just loud enough for everyone nearby to hear. A few students turned, grinning or whispering to each other. Damian’s face burned. He hadn’t even checked his grade yet, but now it felt like everyone knew before he did. For the rest of class, he sat in silence, fuming. The second he got home, he logged into the student portal. Sure enough—a failing score in science. His stomach twisted. Keene failed him .By the time he closed his laptop, the humiliation had boiled into something sharper. If Keene wanted to make him look bad, Damian would make sure the entire school saw him in a worse way. The next morning, the entire projector in Mr. Keene’s classroom flickered on, displaying a slideshow of personal vacation photos. Images of Keene lounging on a beach, posing in sunglasses, and even sipping a massive tropical drink filled the screen. The class exploded with laughter, students pulling out their phones to snap pictures." Where are these coming from?!" Keene stammered, frantically clicking buttons to shut it down. Noah, watching from across the room, immediately knew this wasn’t an accident. A targeted attack like this meant someone had planted an event in the system—a specific trigger that activated when Mr. Keene logged into his account.

if user == “MrKeene”:
 displayPhotos(“Keene_Vacation_2018”)

Noah’s stomach dropped. This was personal.And in the hallway, just out of sight, Damian smirked.

THE CYPHER FILES 3

Paragraph 4: A Messy Indentation Gives It Away (Indentation)

Noah’s eyes narrowed as he scanned the code controlling the projector. Something about it felt familiar. It wasn’t just the way the images had been called—it was how the code was written. In programming, indentation organizes code into neat blocks, making it readable and functional. Without proper indentation, programs become hard to follow and prone to errors. But this wasn’t just sloppy coding—the structure was a mess, just like the hacked smart TV code from before.

if user == “MrKeene”:
displayPhotos(“Keene_Vacation_2018”)
logEvent(“Triggered at login”)

Some lines were pushed too far over, while others weren’t aligned at all. It was almost like someone wasn’t worried about keeping it clean.Noah clenched his jaw."Whoever did this isn’t just playing around anymore," he muttered under his breath.This was the same hacker. The same mistakes. The same signature coding flaws.Glitch was back.And this time, he wasn’t just testing the system—he was using it to humiliate people.

Paragraph 5: Olivia’s First Clue (Variable)

During Olivia’s office aide period, she sat at the front desk, organizing paperwork, when she overheard a heated conversation between two teachers.“It’s not just Keene’s computer,” one of them said. “My gradebook was a mess this morning. I had to redo half of my records.”Olivia’s curiosity immediately kicked in.She slid over to one of the office computers, pretending to check attendance logs while secretly pulling up an internal system report. The report showed a list of recently accessed files—but one entry stood out.A variable name that didn’t belong.

fileAccess = “cipher_to_keene44.tmp”

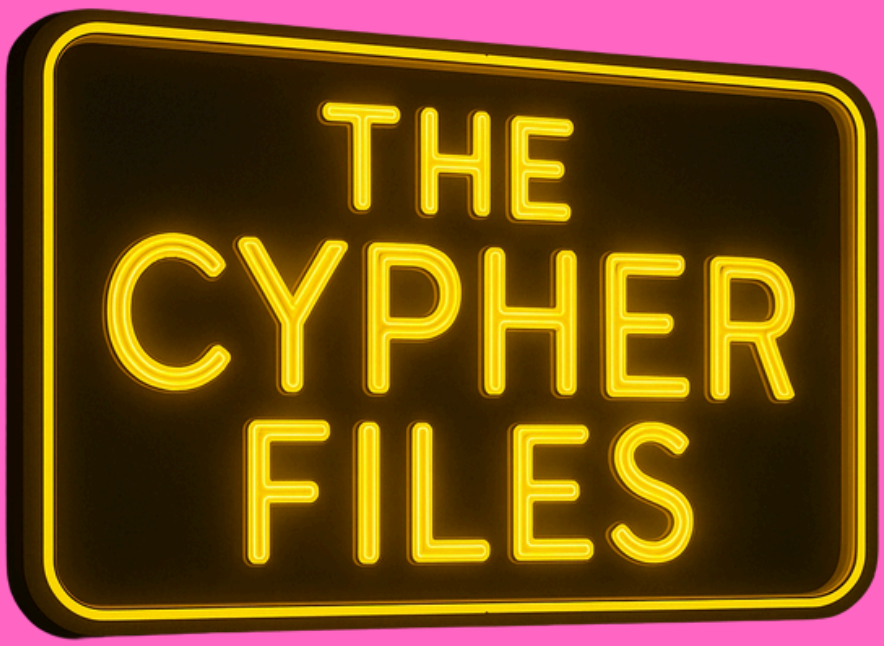
That wasn’t a normal file name. Someone had planted it.She scribbled it onto a sticky note. Noah needed to see this.

Paragraph 6: The Meeting at Lunch (Wrap-Up)

At lunch, Olivia slid the note across the table.“I found this in the system logs.”Noah frowned. “cipher_to_keene44.tmp?”“It was accessed right before the teachers got locked out. It has to mean something.”Noah exhaled, meeting Olivia’s gaze.“We need a plan.”

CHAPTER 3
RECAP

THE CYPHER FILES 4



Chapter 4:
Entry Door
Security Breach

THE CYPHER FILES 4

Paragraph 1: The First Security Breach (User Input)

It started with a rumor. Noah wasn't paying much attention at first—just grabbing his books from his locker—when he overheard a group of students whispering excitedly. "Dude, I swear, I just swiped my ID, and the teacher's door unlocked." Another student scoffed. "No way. It probably just glitched." The first one insisted, "I'm telling you, it worked! A bunch of us got in this morning."

Noah's brow furrowed. That wasn't supposed to happen. The entry doors were restricted to teachers and staff—students weren't even supposed to have the option to scan their IDs. Yet, somehow, a random group of students had walked right through. Minutes later, a hall monitor stopped a few of them near the faculty lounge, demanding to know how they had gotten in. One of them shrugged. "The door just opened. We didn't do anything."

But Noah wasn't buying it. If the system was malfunctioning, it would have been a school-wide issue. But from what he was hearing, only certain students were getting through. Something was off, and Noah intended to find out what.

Paragraph 2: The Access Logs Don't Add Up (String Variable)

Noah didn't waste time. Between classes, he slipped into the computer lab and pulled up the school's access control system. If students were getting through a restricted door, the logs would show who scanned in and when. He expected to see only teacher and staff names. Instead, what he found made his stomach twist. There were student names mixed in with the staff. That wasn't possible.

Each time someone scanned their ID, the system recorded their name and role as a string variable—a piece of text data that classified them as Student, Teacher, or Admin. But in today's logs, some entries didn't match real student accounts. Instead of "Student," their roles were blank or labeled as "Authorized." Noah leaned closer. Someone had altered how the system identified users. Normally, the system would store an entry like this:

userRole = "Student"

But these logs showed something different—the role had been changed:

userRole = "Authorized"

Noah's heart pounded. Someone had bypassed the student restrictions. And he had a good guess who was behind it.

THE CYPHER FILES 4

Paragraph 3: Olivia Puts Noah on the Spot (Conditional Statement)

Noah was still scanning through the access logs, trying to figure out how someone had altered the system, when a voice interrupted his thoughts. “Noah Anderson, report to the office.” He froze. The office? Why?

When he arrived, the principal, Mr. Lawson, was standing with Olivia, who looked way too pleased with herself. “Mr. Anderson,” the principal said, rubbing his temples. “Your friend here tells me you’re good with computers. We’re having an issue with the entry doors allowing students through when they shouldn’t. IT is busy, and we need this fixed before it causes bigger problems.” Noah shot Olivia a look. She gave a sweet, innocent smile. “You’re always fixing things,” she said, nudging him. “Might as well make it official.”

Noah sighed, stepping toward the computer. He wasn’t sure if he should be annoyed or flattered. As he clicked into the system, his suspicions were confirmed. Someone had tampered with the conditional statements controlling access. In programming, a conditional statement determines what happens based on a rule. The entry doors should have been locked unless the user’s role was "Teacher" or "Admin." But the code had been altered:

```
if userRole == "Authorized":
 unlockDoor()
```

Instead of checking if the person was actually a teacher, it now granted access to anyone marked as "Authorized"—a category that shouldn’t even exist. Noah exhaled. Glitch was getting better at covering his tracks.

But before he could dig deeper, Mr. Lawson clapped him on the shoulder. “Just make sure this doesn’t happen again. Faculty has a meeting later, and we don’t want students wandering in.” Olivia smirked. “Also, they ordered pizza, and they’re keeping it locked away so students don’t raid it.” Noah snorted. “So it’s about pizza security now?” “Hey,” Olivia shrugged. “Teachers take their food seriously.”

Noah shook his head. This wasn’t just about pizza—it was about control. And Glitch had too much of it.

THE CYPHER FILES 4

Paragraph 4: Tracking the Manipulated Access (For Loop & Counter Variable)

Fixing the code was one thing—but Noah needed to know how many times this had happened and who was involved. If Glitch had changed the system, there had to be a pattern. He opened the access logs again and started writing a for loop to scan through the entries. A for loop is a command that repeats a set of instructions a specific number of times. In this case, Noah used it to search through every log entry for the past week, looking for students marked as "Authorized."

```
for entry in accessLogs:
 if entry.userRole == "Authorized":
 print(entry.userName, entry.timeStamp)
```

A list of names popped up on the screen. Noah frowned. This wasn't random. The same names appeared over and over—certain students had been scanning in multiple times per day.To confirm the pattern, Noah added a counter variable, which would track how many times each student gained unauthorized access.

```
accessCount = {}
for entry in accessLogs:
 if entry.userRole == "Authorized":
 if entry.userName not in accessCount:
 accessCount[entry.userName] = 1
 else:
 accessCount[entry.userName] += 1
```

The numbers were worse than he expected. Some students had entered through teacher-only doors over 20 times in the past two days.

Olivia peeked over his shoulder. "That's... not good," she said. "No kidding." Noah exhaled. "Someone gave them access, and they figured it out fast." Olivia pointed at one name in the list. "Wait. I know him. He was bragging yesterday about sneaking into the teachers' lounge."

Noah looked back at the screen, his mind working. If they could find out how the students were learning about the hack, they might be able to trace it back to Glitch. For the first time, they had a lead.

CHAPTER 4
RECAP

Chapter 5:

Setting the Trap

Paragraph 1: A Plan Takes Shape (Cipher, Dot Notation)

Noah and Olivia sat in the nearly empty computer lab, the glow of their screens illuminating their determined expressions. The school day had ended, but their work was just beginning. Noah leaned forward, scrolling through lines of code as Olivia watched over his shoulder. “We know Glitch is testing the system,” Noah murmured, his fingers tapping rhythmically on the keyboard. “So let’s give him something to find.” A cipher is a method of encoding information so that it appears scrambled or hidden. In this case, Noah plans to use a digital cipher to disguise the trap as something valuable. To make sure every action is recorded properly, they will use dot notation, which allows them to track specific data interactions in a structured way.

Paragraph 2: Baiting the Hacker (String Concatenation)

Olivia smirked, already seeing where this was going. “You mean a trap? A file that looks valuable enough to bait him in?” Noah nodded. “Exactly. If he thinks he’s hacking something important, he’ll leave a trail.” Olivia crossed her arms, considering. “Alright, but how do we make sure we catch him? He’s smart—if it looks too easy, he’ll suspect something.” Noah grinned. “That’s why we use a cipher. We hide the real trap inside something that looks encrypted. If he tries to crack it, we log everything he does using dot notation to structure the output.” The cipher acted as an encoded message, something Glitch would think contained valuable data but was really just a mechanism to track him. Dot notation would allow them to track exactly which parts of the system he interacted with, ensuring every one of his actions was recorded.

Paragraph 3: A Cipher Inside a Cipher (String Concatenation)

Olivia raised an eyebrow. “So... a cipher inside a cipher?” Noah chuckled. “Something like that. We’ll also use string concatenation to make sure any data Glitch tries to retrieve will still look legitimate, keeping him busy while we track him.” String concatenation is a programming technique that joins multiple strings of text together, allowing them to construct realistic, misleading data inside their trap to make it appear authentic.

Paragraph 4: Constructing the Decoy (Test Variable)

They spent the next hour crafting the perfect decoy: a file named admin_access.tmp, embedded deep within the school's shared network. To the untrained eye, it looked like a cache of encrypted administrator permissions. But behind the scenes, it contained a silent tracker, designed to log Glitch's access, document his methods, and report back to them. "If he tries to read it, we'll know," Noah explained. "If he tries to edit it, we'll see how. Either way, we'll get a lead." The system relies on a test variable, a special placeholder in the code that checks if an action has occurred. In this case, the test variable will determine whether Glitch has accessed or modified the file.

Paragraph 5: A Critical Error (Logical Operator, Test Variable)

Olivia grinned. "This is genius. But let's double-check everything before we set it live." She leaned over Noah's shoulder, scanning the code. As she read through the trap's logic, her excitement faded. Something wasn't right. Frowning, she pointed to a section of the script. "Wait. This won't work." Noah turned to her, confused. "What do you mean?" "The trigger only activates if the file is modified, right?" Olivia asked. Noah nodded. "But what if he just copies it? Or scans it without changing anything? We won't get any data," Olivia continued. "Also, I just noticed a logical operator issue. If Glitch accesses the file in an unexpected way, the test variable might not trigger correctly." Logical operators control the flow of decision-making in programs, ensuring that specific conditions are met before an action occurs. If not structured properly, logical operators can cause the trap to fail by allowing unintended access without triggering the tracker.

Paragraph 6: Rethinking the Plan (Cipher’s First Debug)

Noah’s expression shifted as he realized the flaw in their plan. She was right. If Glitch was careful enough, he could slip in and out without triggering anything. Their whole plan hinged on the assumption that he would edit the file, but that wasn’t a guarantee. Olivia smirked. “Looks like Cipher needs to debug his own trap.” Debugging is the process of identifying and fixing errors in code. Noah now has to refine their plan to ensure the trap works under all conditions.

Paragraph 7: A Name Takes Hold (Cipher’s Identity)

She leaned back, triumphant. “Cipher, huh? I like that. You should keep it.” Noah rolled his eyes, but the nickname stuck. The name fits—Noah specializes in solving and creating ciphers, turning his problem-solving skills into an identity.

CHAPTER 5
RECAP

Chapter 6: Cyphers AI Unexpected behavior

Paragraph 1 – Anomalies in the System (User Input, User)

Noah and Olivia sat in the dimly lit computer lab, their eyes fixed on the lines of code displayed on the screen. The school day was long over, but their work on the Cipher program was still far from complete. What had started as a simple tracking system had turned into something far more complicated. The program was meant to be a straightforward trap for Glitch, but unexpected issues kept appearing. As they tried to debug it, they noticed something else—some of the system's previous glitches weren't just disappearing, they were repeating. It was as if they were responding to something. Noah frowned and began typing. "I think these system responses are being triggered by user input," he said. Olivia leaned in. "You mean the system is reacting to what people type in?" Noah nodded. "Yeah. The program is designed to execute based on what the user enters."

Paragraph 2 – Investigating the Pattern (Input Variable)

Determined to figure out what was going on, Noah started combing through the system logs. His suspicions were confirmed—each repeated glitch was linked to an input variable. "See this?" he pointed at the screen. "The system isn't just breaking at random. It's storing whatever was entered here and using it to determine the next step." Olivia's brows furrowed. "So whoever set this up can keep adjusting it just by changing the input?" Noah exhaled. "Exactly. And that means this isn't just a simple prank. This is something structured—something that's meant to evolve based on what the user does."

Paragraph 3 – The Unexpected Loops (Loop, Integer)

Noah scrolled further down, his expression growing more serious. "Here's the real problem," he muttered. "It's looping." Olivia blinked. "You mean it's repeating itself?" Noah nodded. "Yeah. This part of the code is set to run in a loop. It cycles through a series of commands based on an integer value. The number determines how many times the loop runs before it stops, but in this case..." He trailed off and pointed at the screen. Olivia's eyes widened. "It doesn't stop." Noah nodded grimly. "If no limit is set, the system will just keep running indefinitely."

Paragraph 4 – Unraveling the Problem (Casting, String)

Olivia spotted another issue. “Wait a second. Look here.” She pointed to a section of the code. “It’s trying to use an integer where it should be using a string.” Noah rubbed his forehead. “That could be a big part of the problem. Casting errors happen when a variable is converted to the wrong type. If the system is expecting a number but gets text instead, it might be misinterpreting the input entirely.” Olivia sighed. “So instead of following a clear instruction, it could be treating the wrong kind of data as a command.” Noah’s fingers flew across the keyboard as he muttered, “That explains why we’re getting these unpredictable responses.”

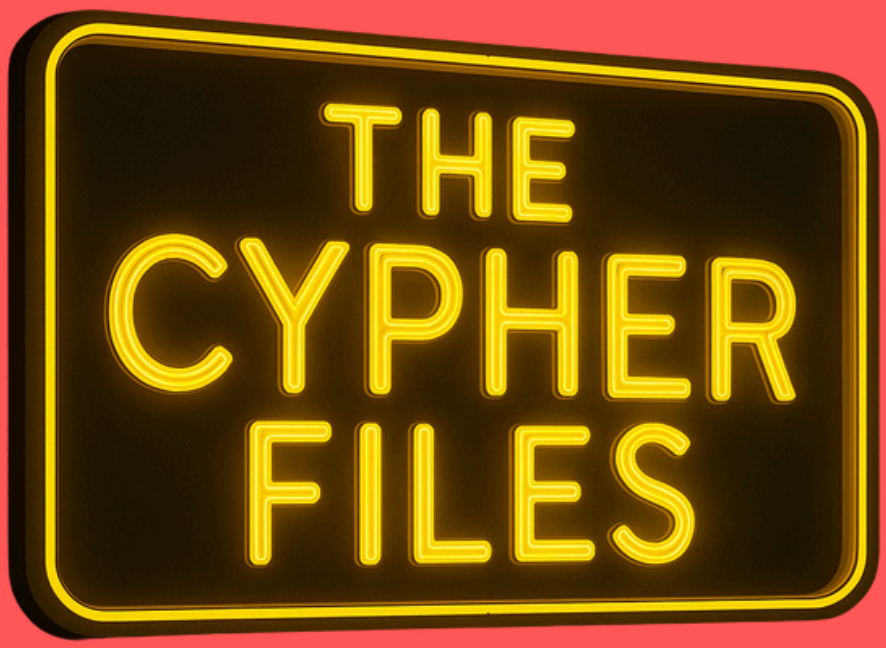
Paragraph 5 – The Bigger Issue (Loop Continues Running)

After making a few changes to the code, Noah sat back and exhaled. “Okay. This should at least help with the input issues.” But as he watched the program run, the loop still didn’t stop. He shook his head. “It’s still running. The loop keeps restarting itself.” Olivia groaned. “So even if we try to shut it down, it might just pick back up the next time someone enters a command?” Noah nodded. “Yeah, and that means we need to find a way to break the cycle completely.”

Paragraph 6 – A Dangerous Realization (Final Cliffhanger)

Olivia leaned back, arms crossed. “So not only do we have to deal with Glitch’s hacks, but we also have to figure out how to stop this thing from running forever.” Noah sighed. “And we still haven’t even finished debugging our own Cipher program. It’s way more complicated than we originally planned.” Olivia smirked. “At this rate, Glitch might beat us to the finish line.” Noah cracked his knuckles. “Not if we figure out how to shut this loop down first.” He took a deep breath. “Because if we don’t, this whole system could keep running forever—whether we want it to or not.”

CHAPTER 6
RECAP



Chapter 7: Cyphers AI vs the User

Paragraph 1 – The Program Starts Responding (User Input, If Statement)

Noah leaned forward, fingers hovering over the keyboard as he ran another test on the Cipher program. Olivia sat beside him, watching intently as the screen flickered to life. Instead of simply tracking data like before, the program now responded to their commands. Noah typed in a basic query, and the system immediately displayed a response. He frowned. “It’s reacting based on user input.” Olivia glanced at him. “So it’s not just running—it’s waiting for us to tell it what to do?” Noah nodded. “Yeah. It’s using an if statement to check what we type and then execute a response.”

Paragraph 2 – Unexpected Behavior (Elif Statement, Else Statement)

Curious, Olivia typed something different. Instead of failing, the system processed her input and returned a completely different message. “That’s weird,” she muttered. Noah studied the code on the screen. “It’s not just using if statements—there are elif statements too. That means it’s checking multiple conditions before choosing what to do next.” Olivia scrolled down. “And look at this. There’s an else statement at the bottom. If nothing matches, the system will always return some kind of response.” Noah exhaled. “So no matter what we enter, it has something ready to say back.”

Paragraph 3 – The System is Testing Them (Test Value, Equality Operator)

Noah decided to push the program further. He entered a new test value to see if it would match a specific response. The program hesitated for a second before displaying a cryptic message. Olivia narrowed her eyes. “Wait… is it testing us?” Noah’s fingers moved rapidly as he traced the program’s logic. “Look at this—it’s using an equality operator to compare our input to something else. It’s not just responding. It’s checking if our answers fit a hidden set of conditions.”

Paragraph 4 – The Problem with Comparisons (Comparison Operator)

Noah tried different inputs, but the system's responses were inconsistent. "It's not just checking for exact matches," he realized. "It's using comparison operators too—greater than, less than, equal to." Olivia's stomach tightened. "So some inputs won't trigger anything unless we unknowingly meet a hidden requirement." She typed in a random number. The screen responded instantly, but when she tried another, the program refused to acknowledge it. "Glitch may have designed this to filter out specific responses," she said. "It's trying to get something from us."

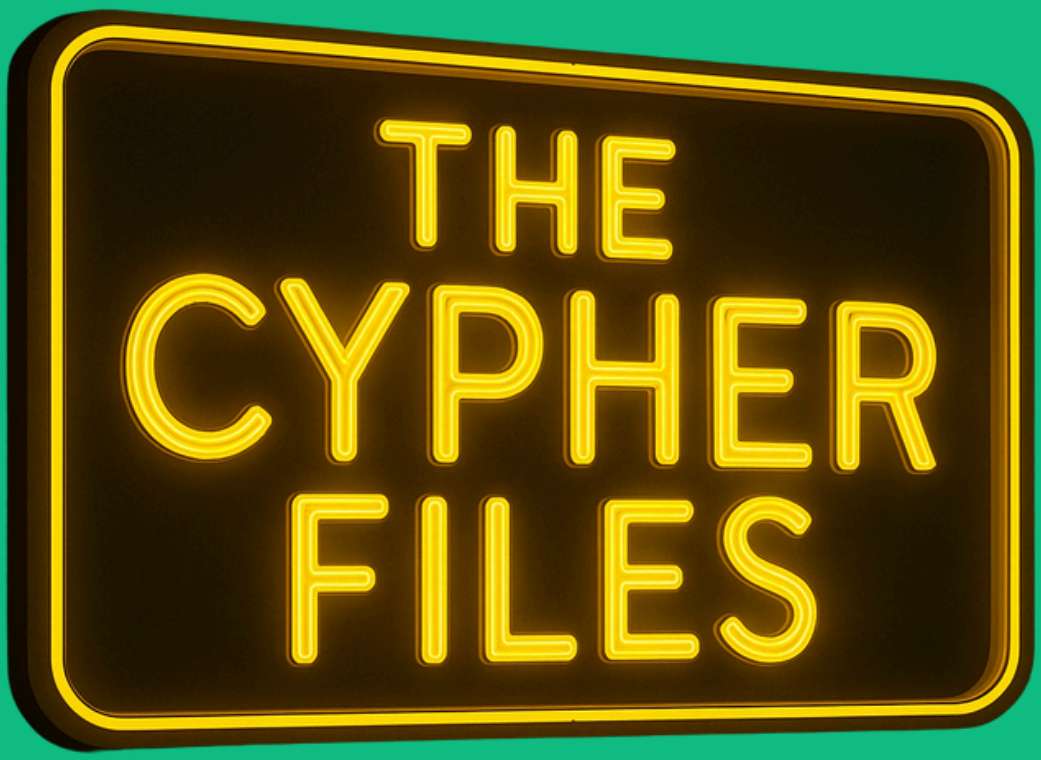
Paragraph 5 – A Trap Within a Trap (Break Statement)

Noah's pulse quickened as he noticed something buried deeper in the code. "Olivia, this isn't just running indefinitely—there's a break statement controlling when the loop ends." Olivia's eyes widened. "So it's stopping itself when a certain condition is met?" Noah nodded. "Yeah, and if we don't enter the right inputs, it might prevent us from shutting it down completely." He leaned back, rubbing his forehead. "This isn't just a bug. This is deliberate." Olivia clenched her fists. "Glitch set this up to lock us out if we don't figure out the right response."

Paragraph 6 – A New Challenge (Final Cliffhanger)

Noah took a deep breath and quickly typed a command, trying to regain control. The screen flickered and then displayed a chilling message: "Invalid Input. Try Again." Olivia's heart pounded. "Noah... what if Glitch is watching?" Noah exhaled slowly. "Then we'd better make sure we don't lose this game."

CHAPTER 7
RECAP



Chapter 8: Cyphers AI Pattern Recognition

Paragraph 1 – A New Discovery (Loop, For Loop)

Noah and Olivia sat side by side, reviewing the logs of the Cipher program. As they scrolled through, Olivia frowned. "Why does the same problem keep happening?" she asked, pointing at a recurring error. Noah's eyes narrowed as he inspected the sequence. "It's because of a loop. The program isn't just glitching randomly; it's repeating actions based on certain conditions." He tapped the keyboard, highlighting a section of the code. "See this? It's using a for loop to run the same process multiple times in a row. That's why we keep seeing the same error."

Paragraph 2 – How the Program is Learning (Counter Variable)

As Noah analyzed further, something else stood out. "This isn't just a simple repetition," he muttered. "Look at this—every time the loop runs, this number increases." Olivia leaned closer. "What is it?" "That's a counter variable. It keeps track of how many times something happens." Olivia raised an eyebrow. "So the program is counting its own mistakes?" "Sort of," Noah admitted. "It's not just running blindly—it's adjusting itself based on how many times it executes."

Paragraph 3 – The Hidden Structure (Range, Indentation)

Olivia noticed another odd pattern. "These errors don't happen all the time, only after a certain number of loops. What's controlling that?" Noah dug deeper into the script. "That's because of the range function," he explained. "It tells the loop how many times to run before stopping—or in this case, before triggering another action." Olivia scanned the screen. "And look at this indentation—it's inconsistent." Noah nodded. "Indentation is how the program knows which commands belong together. If this part isn't aligned properly, it could be why some errors seem out of sync."

Paragraph 4 – The Nested Problem (Nesting, Logical Operator)

Noah’s fingers flew across the keyboard as he traced the code further. "This is bad," he murmured. "Some of these loops are nested inside each other—one loop is controlling another." Olivia’s eyes widened. "That’s why some issues keep happening while others don’t!" Noah pointed to another section. "It’s also using a logical operator to determine when these loops activate." Olivia frowned. "So depending on which condition is met first, different loops will run?" Noah nodded. "Exactly. It’s like the program is deciding which part of itself to execute next based on what’s already happened."

Paragraph 5 – A Pattern Emerges (Increment, A Bigger Problem)

Noah continued scrolling. "This part is even worse," he muttered. "Every time the loop runs, it’s making small adjustments." Olivia studied the code. "That’s an increment function, right?" Noah nodded grimly. "Yeah. The program is gradually shifting its behavior each time it repeats. That’s why it looks like it’s evolving." Olivia’s stomach turned. "So this isn’t just a mistake—it’s intentional. Someone built this to change over time." Noah exhaled. "And if we don’t stop it, it’s only going to get more unpredictable."

Paragraph 6 – A Race Against Time (Final Cliffhanger)

Olivia crossed her arms. "We need to break the pattern before it gets worse." Noah nodded. "The problem is, if we mess with the wrong loop, we could make things worse instead of fixing them." Olivia took a deep breath. "Then we’d better figure out the right one. Fast."

CHAPTER 8
RECAP

Chapter 9: The Encrypted Messages

Paragraph 1 – A Strange Pattern (Loop, For Loop)

Noah and Olivia stared at the screen, their eyes scanning the repeating lines of data in the Cipher program's logs. "These sequences don't match any of the previous errors," Olivia said, puzzled. Noah's fingers moved across the keyboard as he examined the script. "It's a loop," he muttered. "It's running the same process over and over again." He highlighted a section of the code. "It's a for loop—set to repeat a specific number of times. But the question is, what is it looking for?"

Paragraph 2 – Cracking the Code (Break Statement, Counter Variable)

Olivia scrolled further down, pointing at a number that kept increasing. "This counter variable—it goes up every time the loop runs. That means it's tracking something." Noah squinted. "And see this? It doesn't stop at a fixed number. There's a break statement inside the loop. That means the program is waiting for a specific condition to be met before it exits the loop." He shook his head. "This isn't just a standard glitch. This loop is waiting for something specific before stopping."

Paragraph 3 – A Deeper Lock (Comparison Operator, Logical Operator, Encryption)

Noah dug deeper into the code, his jaw tightening. "Here—this is what's controlling when the loop stops." Olivia leaned over. "A comparison operator?" Noah nodded. "Yeah. It's checking something—comparing values. And look here—a logical operator combining two conditions. That means the loop won't exit unless both conditions are met." Olivia's eyes widened. "So it's not just repeating—it's testing inputs against a hidden set of rules." Then she noticed something else—parts of the stored data didn't look like normal text. "Noah... this data looks encrypted."

Paragraph 4 – Finding the Key (Compound Conditional, Indentation, Decryption)
Noah’s eyes flickered toward another section of code, indented slightly from the rest. "Wait—this section isn’t executing yet. That means there’s a compound conditional somewhere." Olivia frowned. "So it’s checking more than one thing before running the next section?" Noah nodded. "Right. And it looks like whatever it’s looking for has to be decrypted first. That’s why it rejects everything we’ve tried so far. The program is looking for a specific key to unlock the message before moving forward."

Paragraph 5 – Unlocking the Message (Decryption, The Big Reveal)
Noah hesitated for a moment before entering a possible decryption key. The screen flickered, then displayed a sudden burst of text: “Message Decoded: Glitch was here.” Olivia’s breath caught in her throat. "This isn’t just a bug—it’s a message he left behind, and he encrypted it on purpose." Noah stared at the screen. "He didn’t just leave these messages hidden in the system. He made sure only someone who understands encryption could find them."

Paragraph 6 – A Chilling Realization (Final Cliffhanger)
Olivia turned to Noah, her expression serious. "If he’s leaving encrypted messages behind, that means he knows someone is tracking him." Noah exhaled slowly. "Not just tracking him. He’s been testing us this whole time." Olivia folded her arms. "Then we’re not just dealing with a hacker. We’re playing his game—and now we know he’s a step ahead." Noah’s eyes darkened. "We need to find the rest of his messages. Fast."

CHAPTER 9
RECAP

Chapter 10:
The Final Move

Paragraph 1 – Cipher’s Program is Finally Running Noah and Olivia stared at the screen as the Cipher program finally launched without errors. After weeks of debugging crashes, refining the tracking algorithm, and ensuring its security protocols were airtight, they had finally overcome every issue. The program now worked seamlessly, ready to trace back every modification Glitch had ever made to the school’s system. Lines of data scrolled rapidly, tracking live system activity and scanning past records for anomalies. "It’s actually working," Olivia whispered in awe. Noah leaned forward, eyes locked on the interface. "Not just working," he corrected. "It’s backtracking through every system change Glitch has ever made. We’re about to expose everything."

Paragraph 2 – Meeting with the Principal

They met with the principal in his office, laying out their case. Olivia took the lead, explaining how Glitch had infiltrated the school’s systems. Noah demonstrated Cipher’s ability to trace back every hack, every login, and every manipulation Glitch had ever done. The principal leaned back, skeptical but intrigued. "Even if you’re right, we can’t accuse a student without undeniable proof," he said. "We need something concrete." Noah and Olivia exchanged a glance before presenting their bold plan.

Paragraph 3 – The Plan: Framing Cipher (Noah) to Catch Glitch

"We need to make Glitch feel like he’s won," Noah explained. "We’ll create a situation where I get blamed for everything. Publicly." The principal raised an eyebrow. Olivia continued, "While Noah is being 'hauled away' by security, we’ll use that distraction to install Cipher on every computer in the lab. The moment Glitch logs in, the program will alert us to his exact location." The principal hesitated before nodding. "If this works, you’ll clear your name. If not..." Noah exhaled. "Then I hope you’ll let me explain before calling my parents."

Paragraph 4 – The Setup: Making Noah Look Guilty

The next day, during a school assembly, the plan went into motion. As students gathered in the gym, a sudden unauthorized system hack disrupted the projector screens. Confused murmurs filled the air as error messages flashed across every screen in the building. Right on cue, the school's security officer stormed into the gym. "We have reason to believe this was an inside job," he announced. His gaze landed on Noah. "Noah Anderson, you need to come with us." Gasps rippled through the crowd as Noah stood frozen, playing his part. Olivia bit her lip, looking convincingly shocked. "No way… Noah wouldn't do this!" she shouted, adding to the act.

Paragraph 5 – The Real Trap is Set As Noah was led out in cuffs, the real operation began. Behind the scenes, Olivia and the principal initiated the mass installation of the Cipher program. Within minutes, it was running on every lab computer, silently waiting. "Now all we need is for Glitch to take the bait," Olivia muttered, refreshing the tracker screen. Noah, meanwhile, sat in the principal's office, waiting for the moment of truth.

Paragraph 6 – The Moment of Truth Then, it happened.

A blinking red alert appeared on Olivia's laptop. "He's in," she whispered. The Cipher program had detected Glitch's student number as he logged into a terminal. The system displayed his exact location—computer 14 in Lab C. The principal's eyes widened. "It worked," he breathed. "We have him."

Paragraph 7 – Glitch is Caught Security officers rushed to the computer lab. Inside, Damian Dalton—Glitch—was hunched over the keyboard, oblivious to his impending downfall. The second he realized people were behind him, his face paled. "What—what's going on?" he stammered. The principal stepped forward, arms crossed. "Damian, we know everything. The Cipher program tracked your every move." Damian's face twisted in panic, his hands hovering uncertainly over the keyboard. "You don't understand," he blurted out. "I was just proving a point! The system was weak—I wasn't trying to ruin anything!" His voice wavered, his confidence cracking as the weight of the evidence bore down on him. He glanced toward the monitor, as if searching for an escape route through the code, but there was none. "You set me up!" he snapped at Noah. "You think you're smarter than me?" The principal stepped forward, unimpressed. "This was your doing, Damian. And now, we have all the proof we need."" A screen behind him displayed a complete history of his hacks, timestamps, and even the backdoor scripts he had inserted into the system. Damian's fingers twitched above the keyboard as if searching for an escape, but there was none.

Paragraph 8 – The Aftermath Noah was cleared of all accusations, his name restored. The principal shook his head, looking at the sheer amount of evidence the Cipher program had gathered. "This... this is remarkable," he admitted. "You two did something no one else could." Olivia grinned. "We just followed the data. Funny, I never thought I'd actually enjoy coding, but after all of this, I see it differently. Coding isn't just about numbers and logic—it's about solving real problems, uncovering the truth. Maybe I was wrong to think it wasn't for me." As they walked out of the office, she smirked at Noah. "You know, Cipher, I think you might have a future in cybersecurity." Noah chuckled. "As long as it doesn't involve getting fake arrested again."

Paragraph 9 – A New Path (Olivia's Return to Programming, Future Goals) As they left the building, Olivia looked thoughtful. "You know," she started, "I think I want to get back into programming." Noah raised an eyebrow. "Really? I thought you said coding wasn't your thing." Olivia shrugged. "It wasn't. But watching what we did—how we solved problems, outsmarted Glitch—I want to be part of that. Maybe cybersecurity isn't just for you." Noah grinned. "So you're saying we should go to college for the same program?" Olivia smirked. "If you think you can keep up with me, Cipher."
Noah laughed as they walked toward the parking lot, the weight of the past few weeks finally lifting. "Well," he said, "I guess we have our next mission." Olivia nodded. "Yeah. And this time, we'll be doing it for real."

CHAPTER 10
RECAP