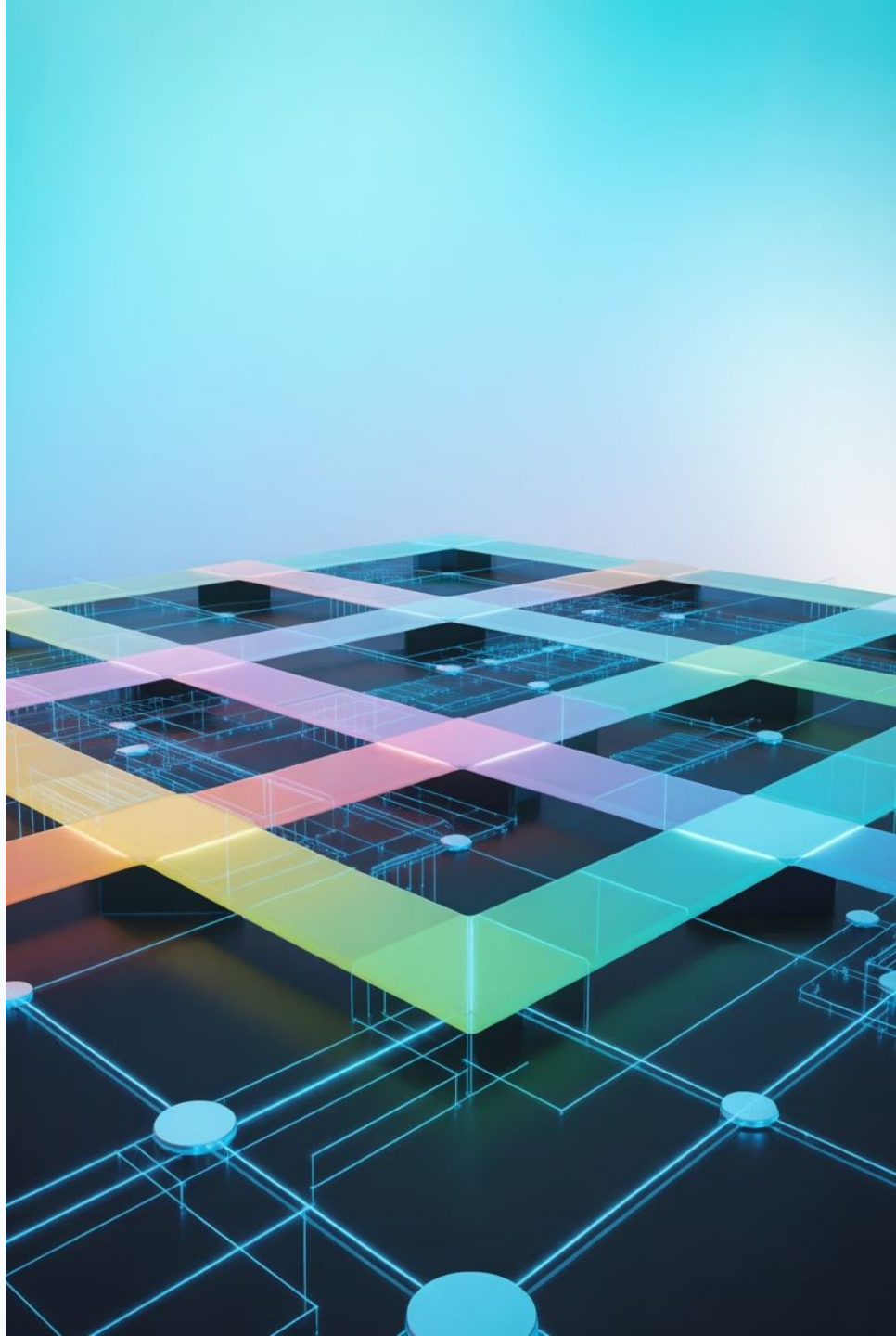
יחידה 1: מבוא לעולם הסייבר

כולל מעבדה מעשית: Hack the 2048 Game



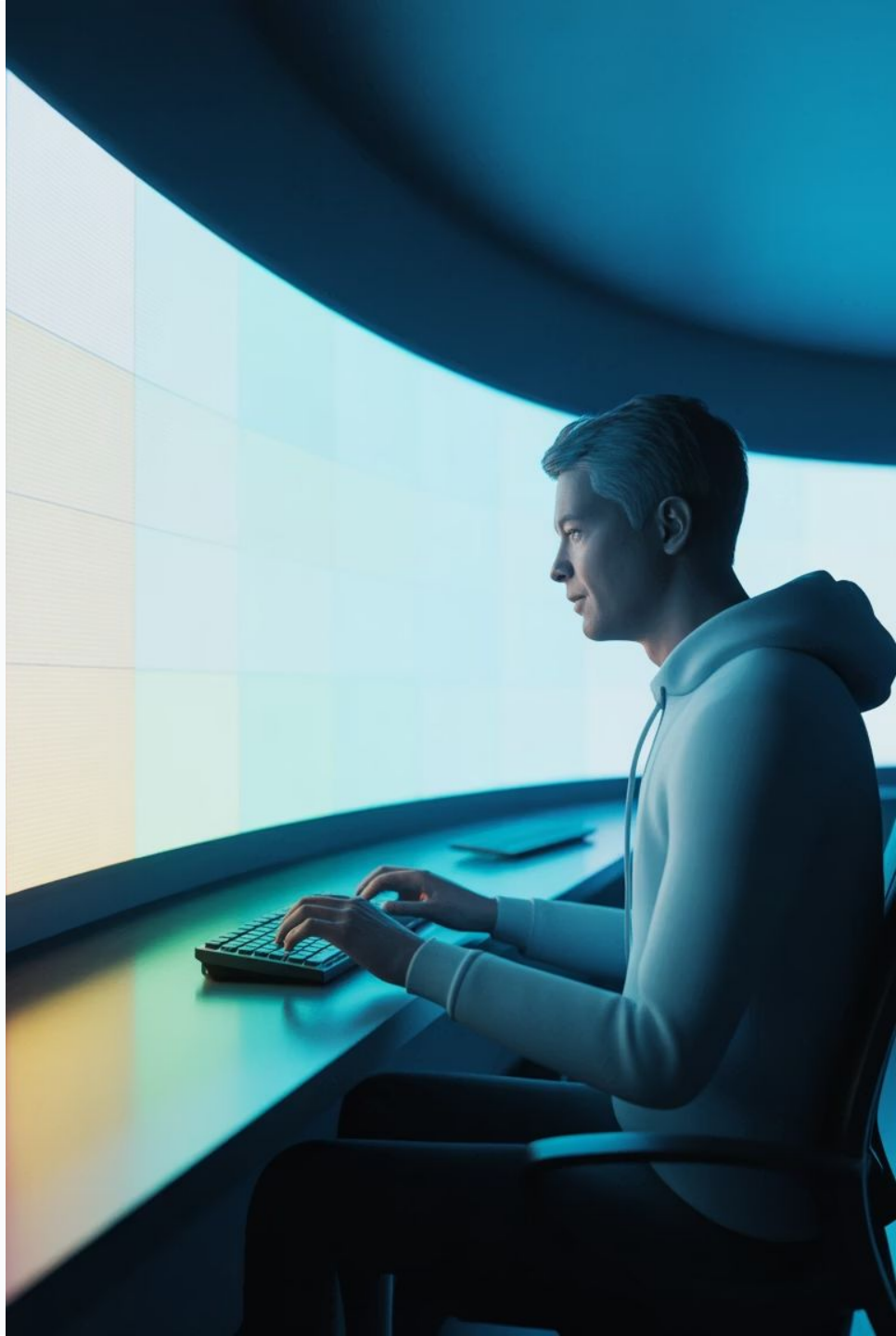
מהות השיעור

חלק ראשון: תיאוריה

שיעור זה מציג את עולם הסייבר, יסודות אבטחת מידע, מושגי יסוד, שחקני איום וחשיבה התקפית-הגנתית.

חלק שני: מעשי

סימולציית תקיפה מלאה על משחק 2048 כדי להבין כיצד מערכת פועלת וכיצד ניתן לנצל חולשה לוגית.



מטרות למידה

להגדיר מהו סייבר ומהי אבטחת מידע

להסביר את מודל CIA

לזהות שחקני איום נפוצים

להבין את החשיבה של תוקף

לזהות חולשה לוגית בקוד

לבצע מניפולציה על משחק 2048

בנוסף, נבין את ההשלכות האבטחתיות של קוד בצד לקוח ונתנסה בכלי מפתחים מקצועיים.

מה זה סייבר?

סייבר הוא תחום רחב העוסק בהגנה על נכסים דיגיטליים חיוניים במרחב הווירטואלי.



נתונים

מידע רגיש ומסווג



מערכות

תשתיות טכנולוגיות



רשתות

תקשורת מאובטחת



משתמשים

זהויות ופרטיות



אפליקציות

תוכנות ושירותים



שרתים

תשתיות ענן

המטרה המרכזית: למנוע גישה לא מורשית, שיבוש, שינוי, מחיקה או גניבה של נכסים דיגיטליים.

דוגמאות לאיומים בעולם האמיתי



פריצות לחשבונות

גניבת סמאות וגישה למידע אישי

גניבת זהות

שימוש בפרטים אישיים למטרות עבריינות

הדלפת מידע

פרסום נתונים רגישים לציבור

כופרות (Ransomware)

הצפנת מידע ודרישת כופר

- תקיפות על אתרי ממשל ומוסדות ציבוריים
- תקיפות על משחקים מקוונים וגניבת חשבונות
- פגיעה בזמינות שירותים קריטיים

CIA Triad

מודל בסיסי לאבטחת מידע

Integrity

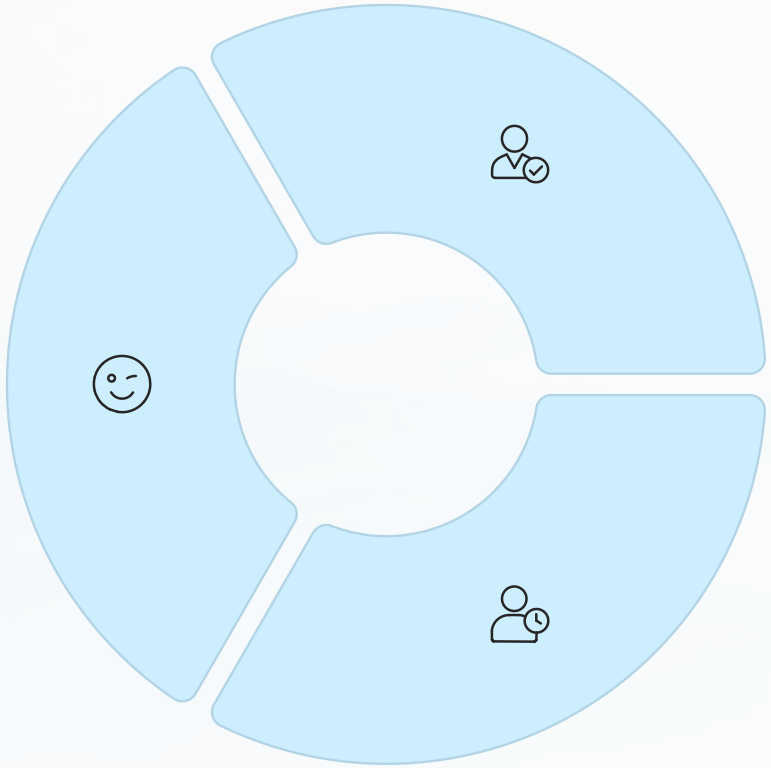
שלמות

שמירה על דיוק ועקביות של נתונים

Availability

זמינות

הבטחת גישה למערכות בעת הצורך



Confidentiality

סודיות

הגנה על מידע רגיש מפני גישה לא מורשית

כל פגיעה באחד משלושת העקרונות מוגדרת כאירוע אבטחת מידע חמור.

דוגמאות פרקטיות לסכנות לפי CIA

סודיות (Confidentiality)

דוגמה: דליפת פסמאות של אלפי משתמשים לרשת האפלה

תוצאות: גישה לא מורשית לחשבונות, גניבת זהות, פגיעה בפרטיות

שלמות (Integrity)

דוגמה: שינוי נתונים רפואיים במערכת בית חולים

תוצאות: טיפול רפואי שגוי, סיכון לחיי אדם, אובדן אמון במערכת

זמינות (Availability)

דוגמה: נפילת מערכות בנק למשך שעות רבות

תוצאות: חוסר יכולת לבצע עסקאות, הפסד כלכלי, פגיעה בשירות

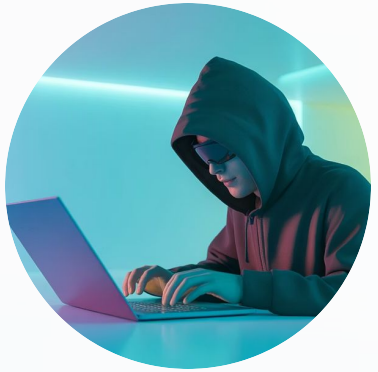
מודל CIA מאפשר לנתח ולסווג כל תקיפה או אירוע אבטחה לפי שלושת העקרונות הללו.

מי תוקף? (Threat Actors)



האקרים צעירים

משתמשים בכלים מוכנים - Script Kiddies



עבריינים

מניעים כלכליים, גניבת מידע ומכירתו



קבוצות מאורגנות

APT - Advanced Persistent Threats



עובדים פנימיים

גישה פנימית למערכות - Insider Threats



ארגוני פשע

כנופיות סייבר בינלאומיות



מדינות

מלחמת סייבר - Nation-State Actors

לכל סוג תוקף יש מטרות, יכולות ומשאבים שונים. הבנת הפרופיל של התוקף חיונית להגנה יעילה.

סוגי מתקפות נפוצות

1

פישינג (Phishing)

מתקפות הנדסה חברתית באמצעות הודעות דוא"ל מזויפות לגניבת אישורים

2

התקפות Web

ניצול פרצות באפליקציות אינטרנט כמו SQL Injection ו-XSS

3

מתקפות רשת

ומתקפות על תשתיות תקשורת DDoS, Man-in-the-Middle

4

נוזקות (Malware)

וירוסים, תולעים, סוסים טרויאניים ותוכנות כופר

5

מניפולציות משתמש

תרמית ושכנוע לביצוע פעולות מסוכנות - Social Engineering

6

פריצות צד לקוח

מניפולציה של קוד JavaScript במשחקים ואפליקציות web

חשיבה התקפית

איך תוקף חושב?

01

מה המטרה שלי?

זיהוי היעד והמוטיבציה

02

מהו החלק החלש במערכת?

סריקה ואיתור נקודות תורפה

03

איזה קוד אחראי על ההתנהגות?

ניתוח טכני של המערכת

04

האם ניתן לעקוף את הכללים?

בדיקת אפשרויות מניפולציה

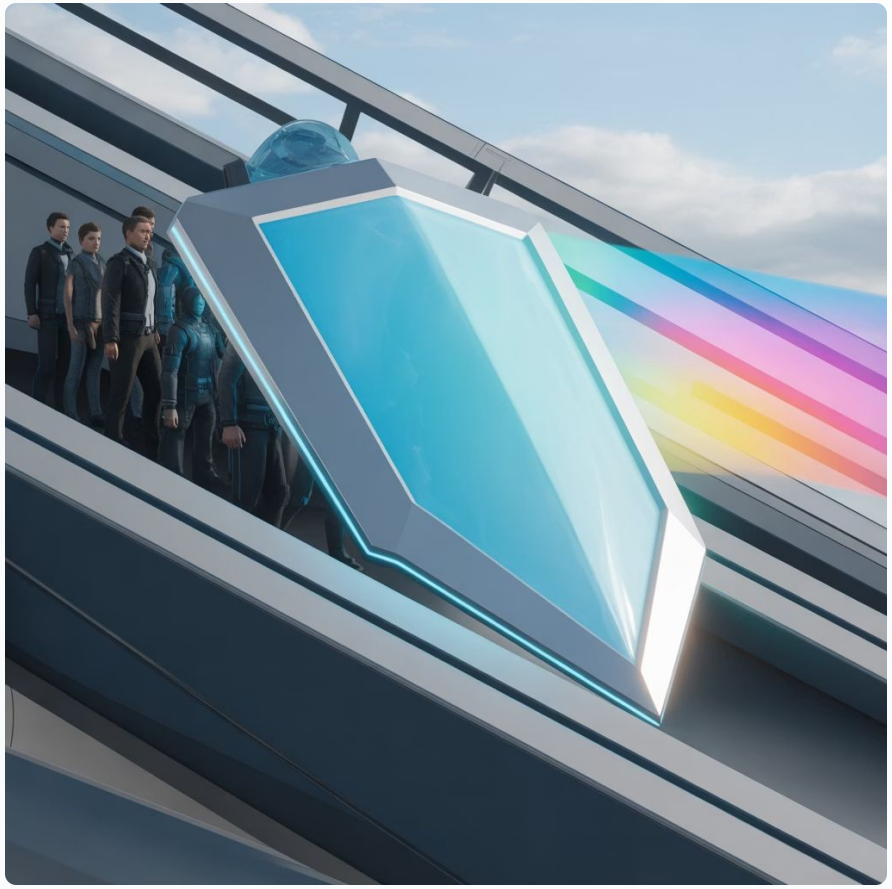
05

מה יקרה אם אשנה ערכים?

ביצוע תקיפה ובחינת תוצאות

> תוקף מחפש תמיד את הדרך הקלה ביותר להשיג את המטרה - הוא חושב **מחוץ לקופסה**.

חשיבה הגנתית



איך מגן חושב?

כיצד המערכת אמורה לפעול?

הגדרת התנהגות רצויה

איך תוקף ינסה לשנות אותה?

חשיבה על תרחישי תקיפה

איפה נקודות החולשה?

מיפוי משטחי תקיפה

האם יש הסתמכות על צד לקוח?

זיהוי חולשות ארכיטקטוניות

האם יש ולידציה בצד שרת?

בדיקת מנגנוני הגנה



חיבור לתקיפת משחקים

משחקים רבים פועלים בסביבת הדפדפן ומבוססים על שלוש טכנולוגיות מרכזיות:

HTML

מבנה הדף והאלמנטים

CSS

עיצוב חזותי וסגנון

JavaScript

לוגיקה, חישובים ואינטראקציה

נקודת תורפה קריטית: כל שינוי בצד הלקוח יכול להשפיע על המשחק אם אין הגנה מתאימה בצד השרת.

זו בדיוק החולשה שננצל במעבדת 2048 שלנו!

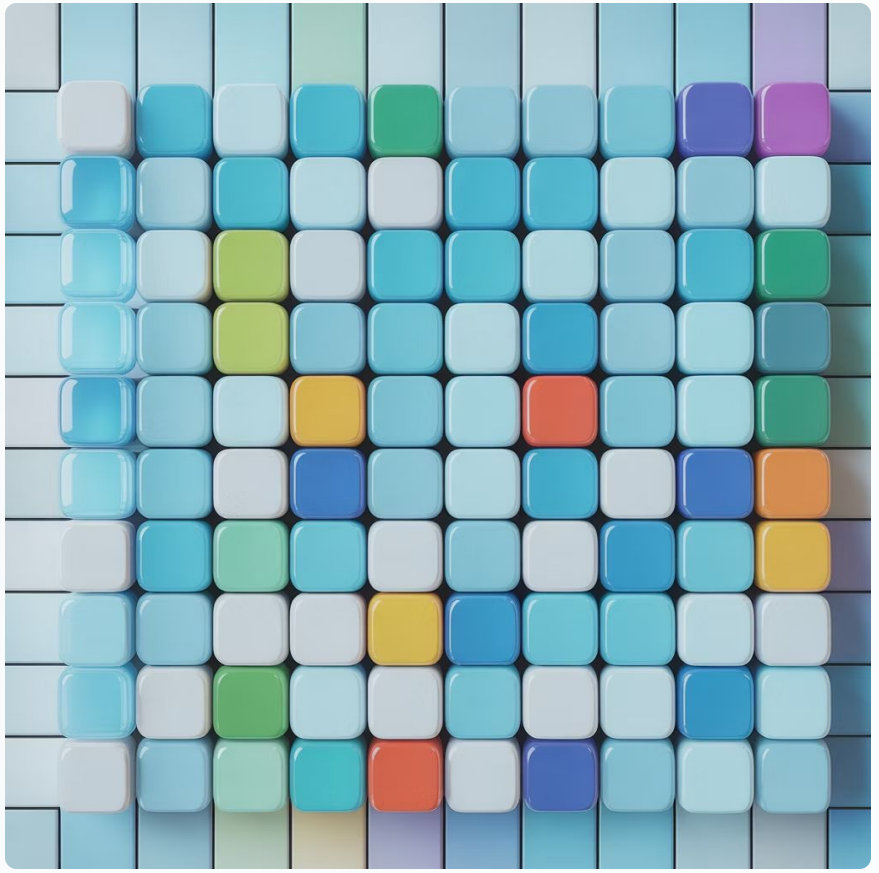
מעבדת 2048

כעת נעבור לחלק המעשי - נשתמש במשחק 2048 המוכר כדי להדגים עקרונות התקפה אמיתיים:

1. **הבנת מבנה הקוד**
2. **זיהוי תנאי ניצחון**
3. **מניפולציה בכלי מפתחים**
4. **חשיבה התקפית מעשית**



מהו משחק 2048?



משחק 2048 הוא משחק פאזל פופולרי המבוסס על חיבור מספרים.

מאפיינים טכניים:

- משחק לוח מבוסס רשת 4×4
- המטרה: לחבר מספרים זהים עד להגעה לאריח 2048
- מבוסס לחלוטין על JavaScript בצד לקוח
- אין תקשורת עם שרת במהלך המשחק
- כל הלוגיקה רצה בדפדפן של המשתמש

חולשה מובנית: המשחק אינו מבצע בדיקות בצד שרת - ולכן פגיע למניפולציות!

מהלך המעבדה – סקירה כללית

פתיחת המשחק
טעינת 2048 בדפדפן

בדיקת מנגנון ניצחון
הבנת חוקי המשחק

פתיחת Developer Tools
גישה לכלי הפיתוח

איתור קבצי JavaScript
מציאת קוד המקור

חיפוש פונקציית ניצחון
זיהוי התנאי הקריטי

שכתוב התנאי
מניפולציה של הלוגיקה

שינוי חוקיות המשחק
עקיפת מגבלות

ניצול בפועל
ניצחון מיידי!

שלב 1: בדיקת המשחק

שאלות לתלמידים:

מה לדעתכם מפעיל את הודעת "You win"?

מה התנאי המתמטי שגורם למשחק להכריז על ניצחון?

איפה בקוד נמצא החישוב הזה?

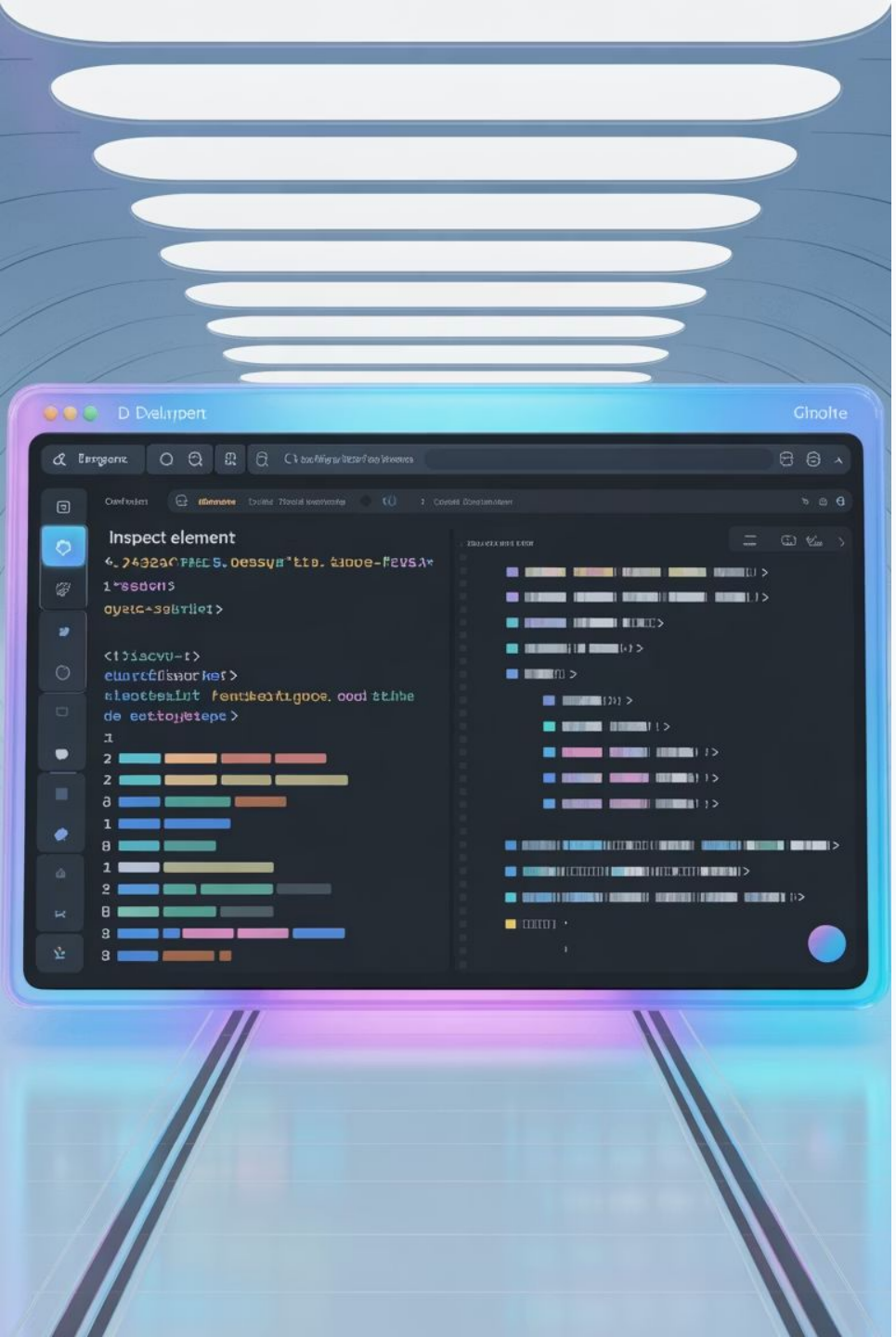
תשובה:

הקוד בודק האם אחד האריחים הגיע לערך 2048. אם כן - המשחק מכריז על ניצחון!

רמזים:

- חישבו על הערך המספרי 2048
- איפה בקוד צריך להיות תנאי בדיקה?
- מי אחראי על זיהוי הניצחון?





שלב 2: פתיחת Developer Tools

כל דפדפן מודרני מכיל כלי מפתחים מתקדמים שמאפשרים לבדוק ולשנות כל אספקט של האתר.

דרך 1

לחיצה ימנית על הדף

בדוק אלמנט / Inspect ←

דרך 2

מקש קיצור

F12

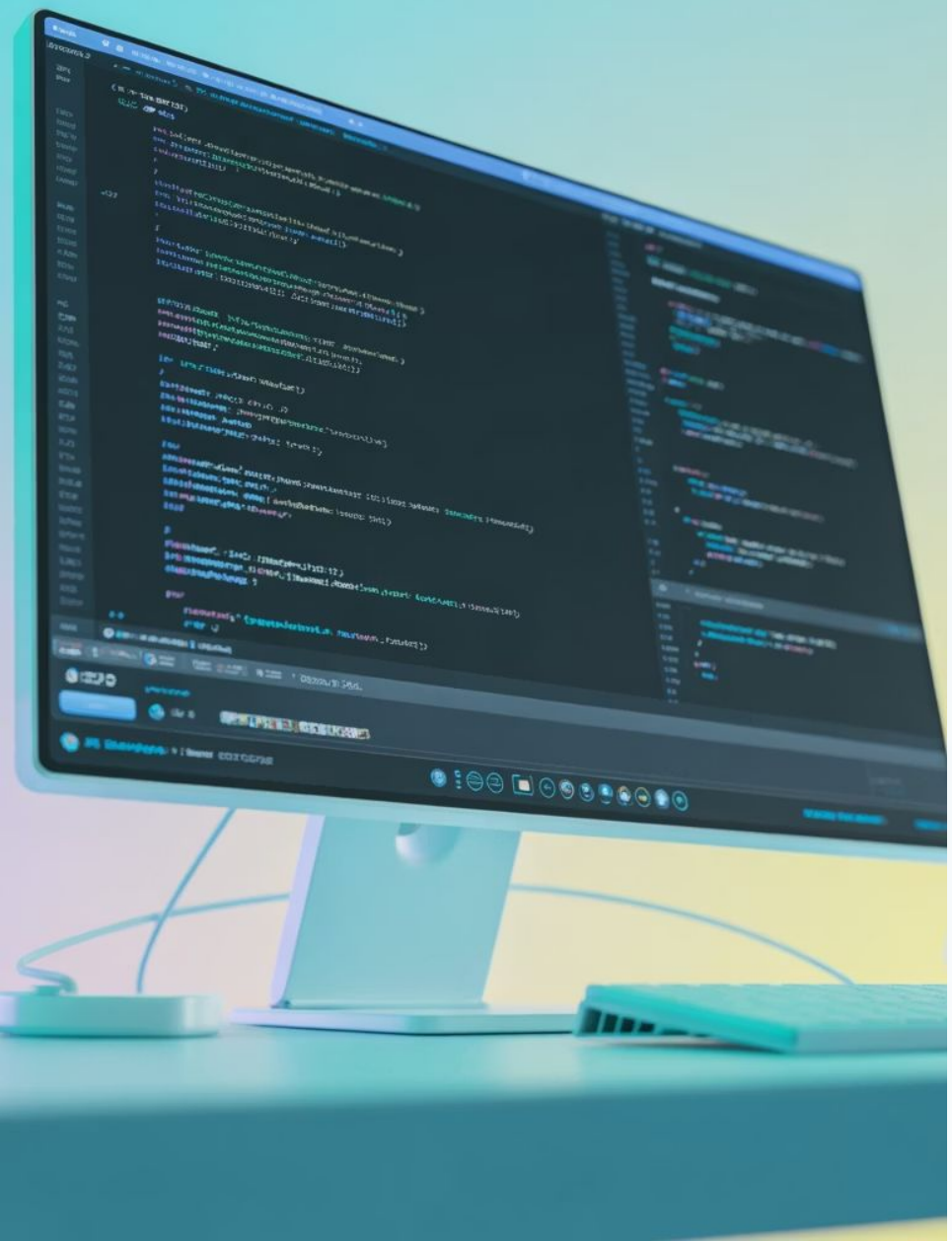
דרך 3

תפריט דפדפן

More Tools → Developer Tools

מה כלי המפתחים מאפשרים לנו לראות?

- קבצים נטענים
- לוגים והודעות מערכת
- קוד HTML, CSS ו-JavaScript
- משתנים ופונקציות



שלב 3: מעבר ללשונית Sources

לשונית **Sources** היא המקום בו נמצא כל קוד המקור של האתר.

1. **לחצו על לשונית Sources**
   תמצאו רשימה של כל קבצי האתר

2. **אתרו את תיקיית /js**
   כאן נמצאים קבצי JavaScript

3. **חפשו את הקבצים הבאים:**
   - `game_manager.js` - ניהול לוגיקת המשחק
   - `html_actuator.js` - עדכון ממשק המשתמש
   - `keyboard_input_manager.js` - טיפול בקלט

הקובץ החשוב ביותר עבורנו הוא `game_manager.js` - כאן נמצאת הלוגיקה המרכזית!



שלב 4: איתור תנאי הניצחון

כעת נצא למסע חיפוש אחר הקוד שקובע מתי ניצחנו במשחק.

פתחו את game_manager.js	חפשו מחרוזות רלוונטיות	מצאו את התנאי

מה לחפש?

קוד שתמצאו:

```
if (tile.value === 2048) {
  this.won = true;
}
```

מחרוזות לחיפוש:

"2048"

"win"

"won"

"victory"

> זהו הקוד הקריטי! הוא בודק אם ערך האריח שווה ל-2048, ואם כן - קובע שניצחנו.

הסבר על חולשה לוגית

גילינו חולשה אבטחתית חמורה בעיצוב המערכת:

הבעיה

המערכת מסתמכת על **בדיקה בצד הלקוח בלבד**

אין אימות או ולידציה בצד השרת

ההשלכה

מי ששולט בדפדפן ← שולט בתוצאה

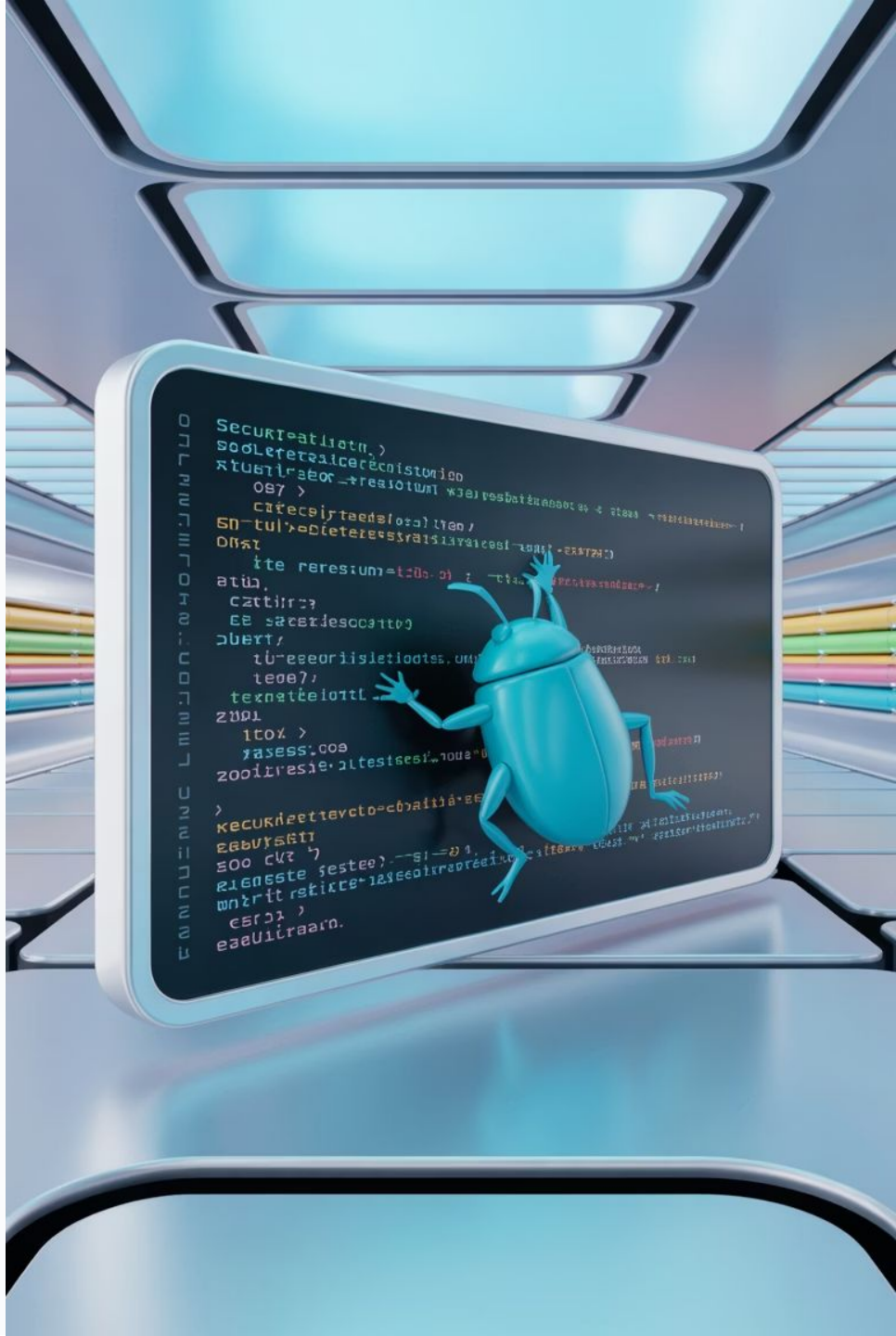
תוקף יכול לשנות כל תנאי לוגי כרצונו

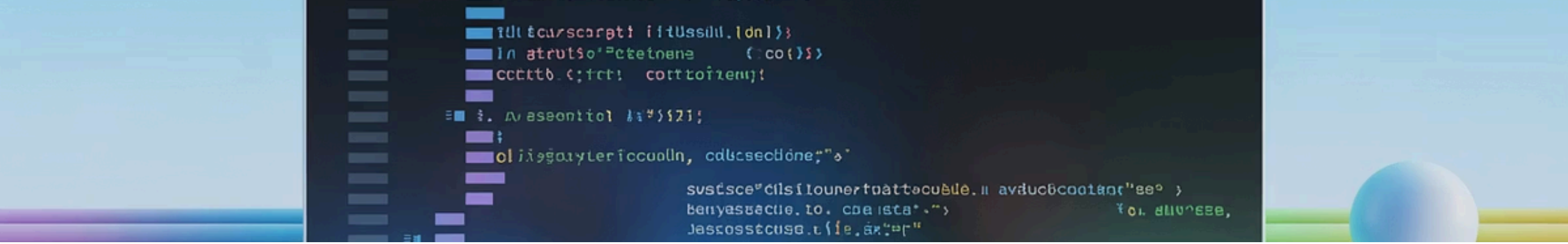
הסיכון

אפשר לעקוף לחלוטין את חוקי המשחק

אין דרך למערכת לזהות מניפולציה

זוהי דוגמה קלאסית לחולשה ארכיטקטונית - העיצוב עצמו פגום מהיסוד!





שלב 5: שינוי ערך היעד

כעת נבצע את התקיפה בפועל!

נעבור ללשונית **Console** ונריץ קוד JavaScript ישירות במערכת.

פקודה ראשונה

```
tile.value = 2048
```

משנה את ערך האריח הנוכחי ל-2048

פקודה חלופית

```
tile.value = 4096
```

יוצר אריח בערך גבוה יותר

התוצאה

שינוי הערך **ללא קשר למהלך המשחק**

עקיפה מלאה של הלוגיקה!

> הצלחנו למנוע את הכללים הבסיסיים של המשחק בקלות מפתיעה.

שלב 6: עקיפת מנגנון Game Over

מעבר לשינוי ערכים, נוכל גם לנטרל לחלוטין את תנאי הסיום של המשחק.



התוצאה:

- המשחק לעולם לא יסתיים

הפקודה:

```
GameManager.prototype.isGameTerminated =
function() {
  return false
}
```

מה זה עושה?

משכתב את הפונקציה שבודקת אם המשחק הסתיים
גורם לה להחזיר תמיד false



שלב 7: ניצחון מיידי

כעת נבצע את התקיפה המרשימה ביותר - ניצחון מיידי ללא משחק בכלל!

אופציה 1: שינוי מצב

```
GameManager.prototype.won = true
```

קובע ישירות שניצחנו

אופציה 2: סיום כפוי

```
GameManager.prototype.isGame = Terminated
function() { return true }
```

גורם למשחק לחשוב שהסתיים בניצחון

אופציה 3: קריאה ישירה

```
this.actuate()
```

מפעיל את עדכון הממשק להצגת ניצחון

בום! 💥 הצלחנו "לנצח" את המשחק מבלי לשחק אפילו מהלך אחד!

שלב 8: הסבר עקרוני
מה למדנו מהמעבדה?

קוד JavaScript ניתן בקלות לשינוי בצד הלקוח

כל מה שרץ בדפדפן נגיש למשתמש ולמניפולציה

משחקים רבים אינם מאובטחים כראוי

מפתחים מסתמכים על "אמון" במקום על אימות טכני

כך תוקפים "מרמים" במשחקים מקוונים

זהו בדיוק העיקרון מאחורי צ'יטים ומודים לא חוקיים

העיקרון זהה באפליקציות קריטיות

אותה חולשה יכולה להיות קטלנית במערכות בנקאות, בריאות או ממשל



חשיבות ממשק ה-Console

ה-**Console** הוא כלי עוצמתי המאפשר:

הרצת קוד

ביצוע פקודות JavaScript בזמן אמת

בדיקת פונקציות

קריאה ובחינה של פונקציות קיימות

בדיקת משתנים

צפייה בערכים שמערכת מחזיקה

שינוי ערכים

מניפולציה של מצב המערכת



זהו כלי מרכזי הן של תוקפים והן של חוקרי אבטחה לגיטימיים.

הקבלה לעולם האמיתי

תרחיש מפחיד: מערכת בנקאית פגיעה

מה היה קורה אם מערכת בנקאית הייתה מסתמכת על צד לקוח בלבד?

שינוי יתרה

תוקף יכול היה לשנות את היתרה בחשבון שלו מ-100₪ ל-1,000,000₪

שינוי הרשאות

משתמש רגיל יכול היה להעניק לעצמו הרשאות מנהל

עקיפת אימות

אפשר היה לעקוף לחלוטין בדיקות זהות וסמאות

העברות בלתי חוקיות

ביצוע העברות כספים ללא אישור או גבולות

המסקנה: צד שרת חייב תמיד לבצע בדיקות, ולידציות ואימות עצמאי!



טעויות נפוצות של מפתחים

Common Security Mistakes

הסתמכות על JS בלבד

הנחה שהמשתמש לא ישנה את הקוד בדפדפן

סיכון: גבוה מאוד

בדיקות לוגיות בדפדפן

ולידציה של קלט רק בצד לקוח

סיכון: מאוד חמור

חוסר ולידציה בשרת

אי-אימות של נתונים המגיעים מהלקוח

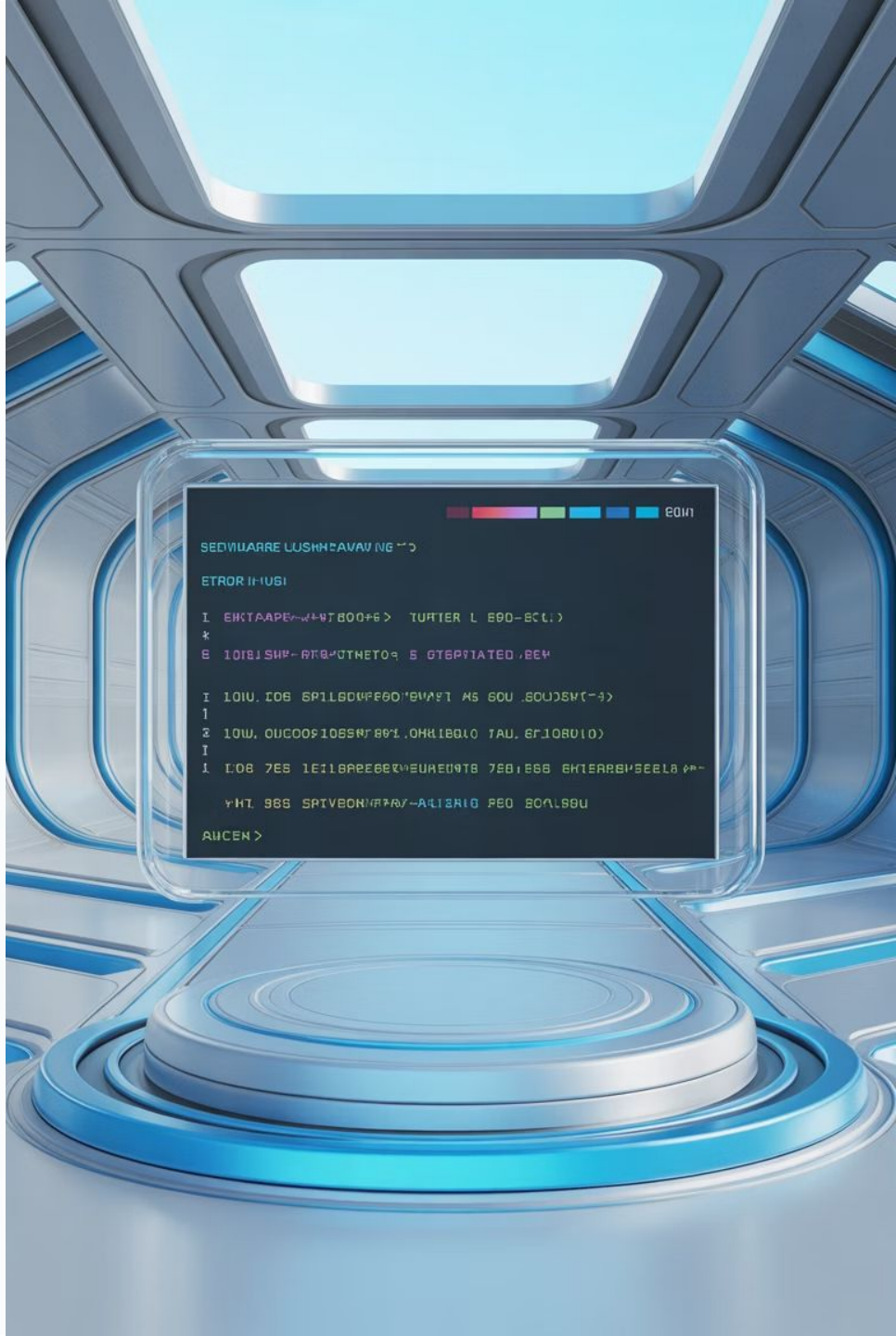
סיכון: קריטי

שליחת פרמטרים רגישים

העברת מידע קריטי ב-Client Side

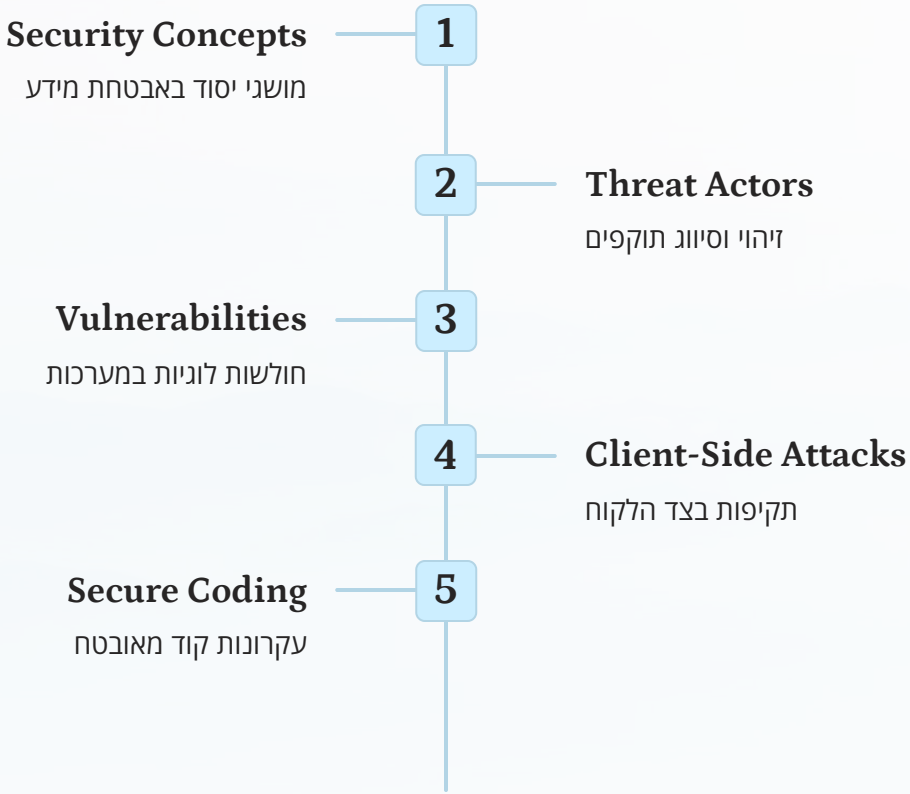
סיכון: חמור ביותר

כלל ברזל: Never Trust the Client! לעולם אל תסמכו על צד הלקוח.



חיבור ל-LPI Security Essentials

המעבדה שביצענו מכסה נושאים מרכזיים מתוך בחינת LPI Security Essentials:



הידע שרכשתם כאן הוא בסיס חיוני להבנת אבטחת מידע מודרנית ולהצלחה בבחינה.



שאלות לדיון

Questions for Discussion

האם משחקים צריכים להיות מאובטחים?

האם זה באמת משנה אם מישהו מרמה במשחק סינגל-פלייר? מה לגבי משחקים מולטי-פלייר?

אילו בדיקות צריך לבצע בצד שרת?

איזה מידע חייב להיות מאומת? מה ניתן לסמוך עליו מצד הלקוח?

איך הייתם מגינים על משחק 2048?

אילו שינויים ארכיטקטוניים נדרשים? האם זה בכלל אפשרי בלי שרת?

מה ההשלכות על מערכות גדולות יותר?

איך אותם עקרונות חלים על בנקים, בתי חולים או רשתות חברתיות?

סיכום ולקראת השיעור הבא

מה למדנו היום?

יסודות סייבר

הכרנו את עולם אבטחת המידע, מודל CIA ושחקני איום

חשיבה התקפית

הבנו כיצד תוקף מנתח ומזהה חולשות במערכות

מניפולציה מעשית

ביצענו תקיפה אמיתית על משחק 2048 באמצעות כלי מפתחים

ניתוח חולשה

ניתחנו חולשה לוגית אמיתית והבנו את השלכותיה

בשיעור הבא:

נעמיק בעולם החשיבה ההתקפית והגנתית (Red Team vs Blue Team) ונלמד על **מחזור חיי התקיפה** (Cyber Kill Chain).

תודה!

שאלות?

> *"The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards."*
>
> — Gene Spafford, Computer Security Expert

צרו קשר למשוב, שאלות או דיון נוסף בנושאי אבטחת מידע.

