AWS Security Agent — אבטחת אפליקציות פרואקטיבית ופנטסט אוטונומי

AWS Summit Tel Aviv 2026 ,סיכום סשן — SEC303

Itay Meller, Senior Security SA, AWS · May Peretz Mulla, Technical Account Manager, AWS

10 בספטמבר 2026 | משך: 35 דקות | הופק מהקלטת הסשן

מסלול: AWS AI Cloud Ops, Infrastructure & Security | רמה: 300 (Advanced)

על המסמך הזה

המסמך מסכם את הסשן SEC303 מתוך AWS Summit Tel Aviv 2026. הסשן הוצג בשני חלקים: איתי מלר, Senior Security SA ב-AWS, הסביר איך AWS Security Agent בנוי מבפנים — ארכיטקטורת הסוכנים, מנגנוני הבטיחות והוולידציה; מאי פרץ מולא, Technical Account Manager ב-AWS, הראתה איך כל זה נראה למשתמש בקונסול, וסיימה בהכרזה על AWS Continuum.

המסמך נבנה מתמלול אוטומטי של ההקלטה ומהסליידים שחולצו מהווידאו עצמו. חותמות הזמן בגוף המסמך מפנות לנקודה המדויקת בהקלטה.

איך לקרוא את זה

- **פרק 1 — תקציר מנהלים:** חמש דקות קריאה. מה הוכרז, מה זמין היום, ומה עדיין בפריוויו.
- **פרקים 2–3 — למה ומה:** פער המהירות בין פיתוח לאבטחה, ומה השירות מציע בפועל.
- **פרקים 4–7 — מתחת למכסה המנוע:** חמשת שלבי הריצה, יחידת העבודה הבסיסית, שכבות הבטיחות ומנגנון הוולידציה. זה הלב הטכני של הסשן.
- **פרקים 8–10 — חוויית המשתמש:** Agent Spaces, דרישות אבטחה, Design Review, Threat Modeling, Code Review ו-Pentest.
- **פרקים 11–12 — Continuum ומה לוקחים:** לאן השירות הולך, ומה כדאי לעשות עכשיו.

הערת מקור המסמך הופק מתמלול אוטומטי של ההקלטה. התמלול משבש באופן שיטתי מונחים לועזיים ואף מילים עבריות — למשל "הבטחה" במקום "אבטחה" ו"שריקה" במקום "סריקה". הציטוטים במסמך מובאים כלשונם מהתמלול ואומתו מול ההקלטה בחותמת הזמן המצוינת; בגוף הטקסט המונחים מנורמלים לצורתם התקינה. כל המספרים והשמות שמופיעים כאן לקוחים מהסליידים או מהדברים שנאמרו על הבמה.

1. תקציר מנהלים

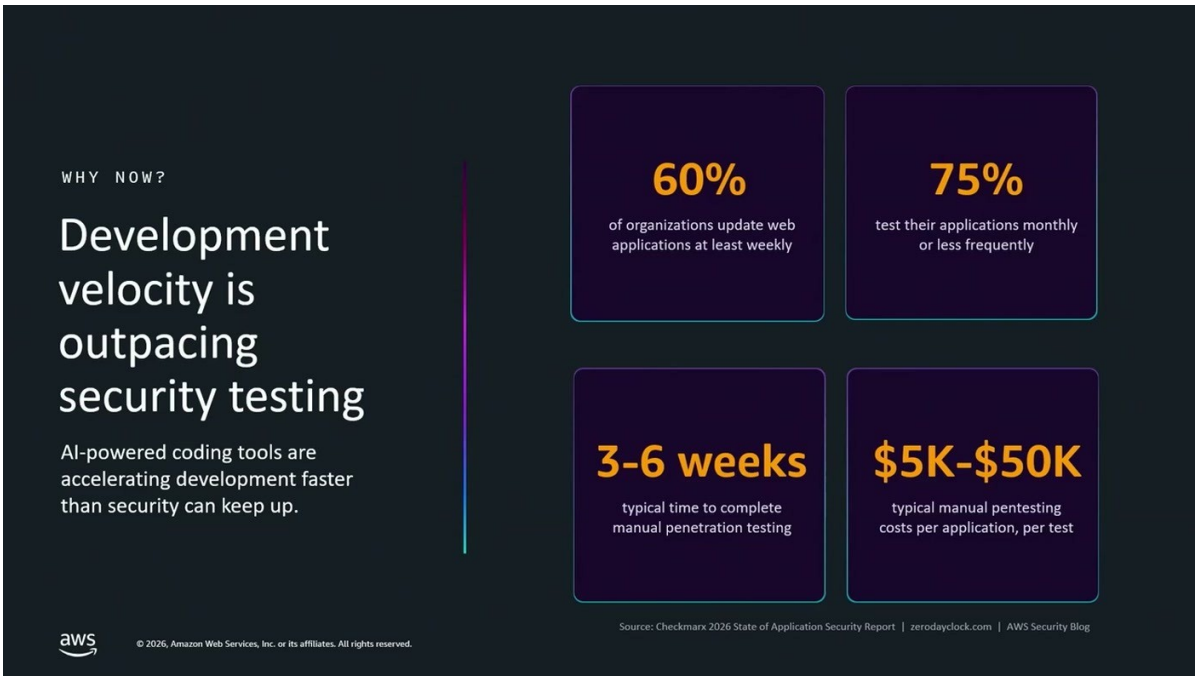
הסשן עסק בפער אחד: כתיבת קוד האיצה פי עשרה עד פי מאה בזכות כלי AI, בעוד בדיקות האבטחה נשארו באותו קצב. AWS Security Agent, שהוכרז ב-re:Invent האחרון וזמין היום, הוא הניסיון של AWS לסגור את הפער באמצעות סוכן AI שמבצע design review, threat modeling, code review ופנטסט אוטונומי.

- **הבעיה היא אסימטריה, לא חוסר כלים.** לפי הנתונים שהוצגו (מקור על הסלייד: Checkmarx 2026 State of Application Security Report), 60% מהארגונים מעדכנים אפליקציות ווב לפחות פעם בשבוע, אך 75% בודקים אותן פעם בחודש או פחות. פנטסט ידני אורך 3–6 שבועות ועולה 5,000–50,000 דולר לאפליקציה לבדיקה.
- **זה pipeline אחד, לא עוד סורק.** השירות מאחד SAST, DAST ו-SCA עם exploration אוטונומי מבוסס AI — מה-login, דרך הסריקה וההרצה, ועד ניקוי המשאבים שנוצרו.
- **הערך המרכזי הוא שרשור פגיעויות להוכחת ניצול.** היכולת לחבר כמה חולשות לאקספלויט אחד היא מה שמבדיל רעש מבעיה אמיתית — אם יש אקספלויט, זה משהו שרוצים לטפל בו מיד.
- **בטיחות נבנתה בארבע שכבות, שתיים מהן דטרמיניסטיות.** בידוד רשת דרך פרוקסי, מדיניות אישור ב-Cedar על כל קריאת כלי, ומעליהן הערכת LLM ו-content guardrails. שתי השכבות התחתונות אינן מערבות LLM בהחלטה.
- **הממצאים עוברים ולידציה לפני שהם מגיעים אליכם.** כל ממצא עובר pipeline של שחזור, assertions שהוגדרו מראש וניקוד ביטחון; ממצא שהוולידטור הצליח להפריך מסומן כ-false positive.
- **AWS Continuum הוא הצעד הבא, וההשקעה נשמרת.** Security Agent הפך לחלק מ-Continuum, שמעביר את המודל מהרצה on-demand להרצה מתמשכת. Continuum Code Vulnerabilities נמצא בגייטד פריוויו; לא נדרשת מיגרציה — אותו דאטה, אותו קונסול.

2. הבעיה: קצב הפיתוח עוקף את קצב הבדיקה

איתי מלר פתח בטענה שהזמן היקר בהנדסת תוכנה מעולם לא היה בכתיבת הקוד עצמו אלא בהנדסתו למערכת מתפקדת — וששלב זה הוא שהואץ דרמטית. התוצאה היא אסימטריה: צד אחד של המשוואה רץ, השני לא.

> **[1:35] —** כתיבת הקוד הפכה הרבה יותר מהירה מה שלא הפכה הרבה יותר מהיר זה הבטחת הקוד והאסימטריה הזאת מייצרת מצב שהוא לא ססטיינבל



ארבעת המספרים שהוצגו כהצדקה לשירות — תדירות עדכון מול תדירות בדיקה, ולצידם משך ועלות של פנטסט ידני

כשהמרצה שאל את הקהל מי מעדכן אפליקציות לפחות פעם בשבוע, עלו שתי ידיים — והוא השתמש בכך כדי להדגיש את הפער בין הסטטיסטיקה לתחושה בחדר.

> **[1:35] —** מישהו פה מעתקן אפליקציות לפחות פעם בשבוע? שניים, זה לא 60%.

ההסבר לפער הוא כלכלי ולא טכנולוגי: בדיקות אבטחה יקרות, לוקחות זמן ומעכבות את תהליך הפיתוח. זו, לדבריו, בדיוק הסיבה שבגללה AWS השיקה את Security Agent ב-re:Invent האחרון.

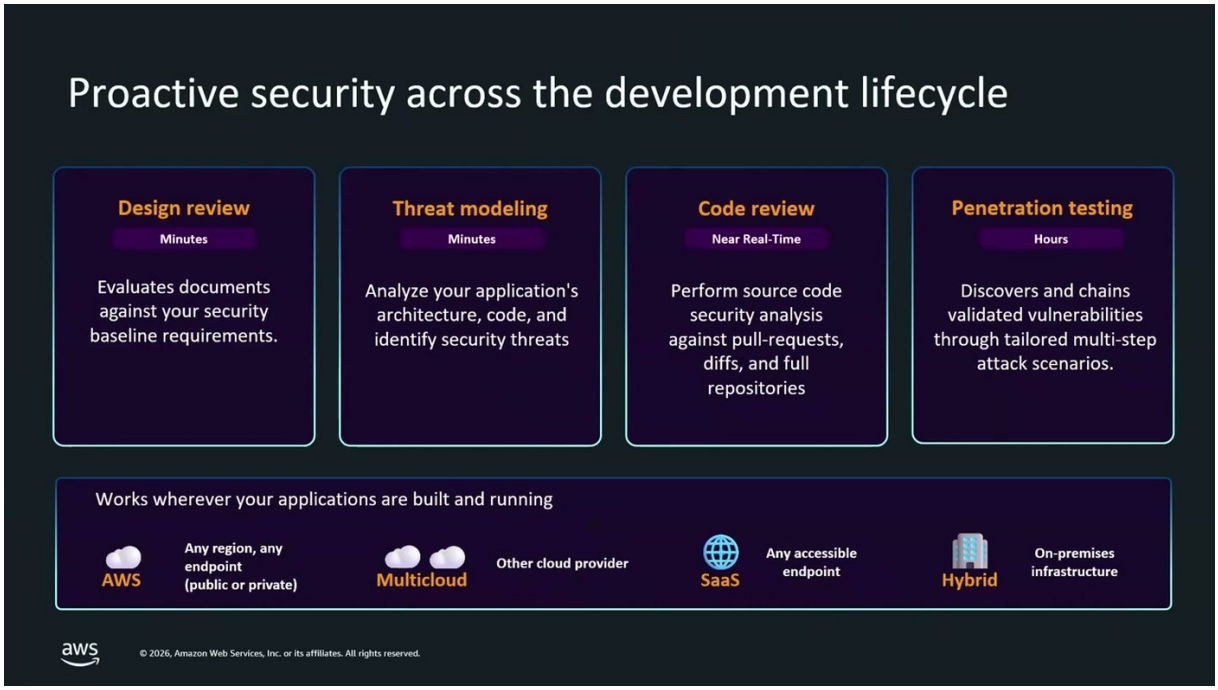
3. מה השירות עושה

התשובה לשאלה "למה AI" הורכבה משני חלקים. הראשון: מודלים עובדים היטב עם קונטקסט לא-מובנה — הקוד, ה-Infrastructure as Code, דיאגרמות ארכיטקטורה, מסמכי Swagger שמתארים את ה-API. כל קונטקסט שמזינים לסוכן הופך את הפנטסט לאפקטיבי יותר. השני: ההתפתחות ביכולות המודלים בשנה האחרונה מאפשרת למצוא חולשות שהיו לוקחות לחוקר מיומן שבועות או חודשים — חולשות שמערבות כמה חתיכות קוד או כמה פגיעויות שונות.

> **[3:07] —** היכולת הזאת לעשות צ'יינינג לכמה פגיעויות שונות, ולייצר הדגמה, לייצר אקספלויט, זו בעצם יכולת חזקה מאוד. כי היא נותנת לצפתי הבטחה את היכולת להבדיל בין רעש לבין בעיה אמיתית

שלושת עמודי התווך שהוצגו: איחוד SAST+DAST+SCA עם פנטסט אוטונומי, מודעות לקונטקסט, וממצאים מאומתים

מעבר לפנטסט, השירות מציע סוויטה של יכולות: סריקת ריפוזיטורי שלם, סריקת Pull Request ספציפי, feedback loop ישירות מה-IDE עוד לפני שנוצר ה-PR, וכן design review ו-threat modeling לארגונים שמבצעים תהליכים כאלה לפני תחילת הפיתוח.



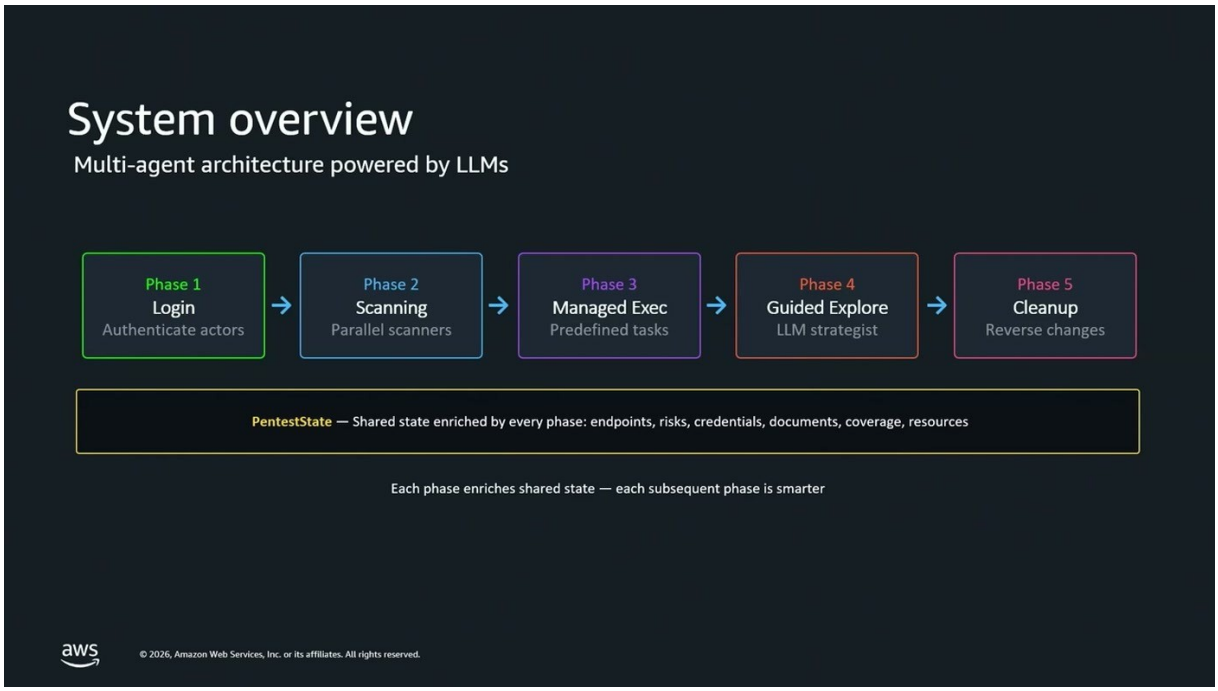
ארבעת המודולים לאורך מחזור החיים, עם זמני הריצה הצפויים שלהם, ומתחתם סביבות ההרצה הנתמכות

נקודה שחזרה על עצמה: מיקום האפליקציה אינו קריטריון. הדרישה היחידה היא endpoint נגיש והוכחת בעלות על הדומיין.

[4:37] — כל מה שהאג'נט צריך בשביל להתחיל לבצע את הבדיקה זה בעצם end point

4. הארכיטקטורה: חמישה שלבים ו-state משותף

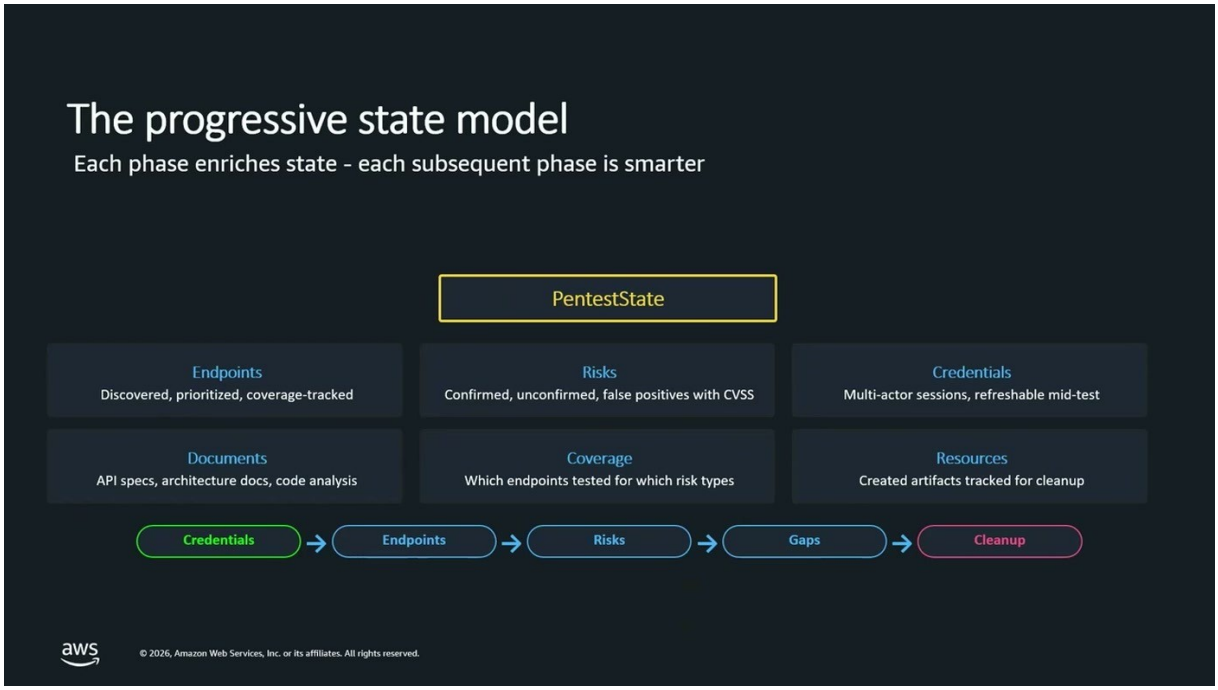
[4:37] — אז המערכת פועלת בחמישה שלבים. הזדהות, שריקה, הרצה, מחקר וניקוי.



חמשת השלבים — Login, Scanning, Managed Exec, Guided Explore, Cleanup — ומתחתם ה-PentestState שמחבר ביניהם

בשלב ההזדהות הסוכן מתחבר לאפליקציה בעצמו: הוא מפעיל headless browser, ממלא שם משתמש וסיסמה, מטפל ב-OTP וב-redirects, ושומר את הטוקנים ועוגיות ה-session. כל אלה נשמרים ב-shared state.

[6:12] — הסטייט המשותף בעצם הוא הרכיב שמחבר את כל השלבים שראינו יחד



ששת התחומים שה-PentestState מחזיק, והסדר שבו הם מתמלאים לאורך הריצה

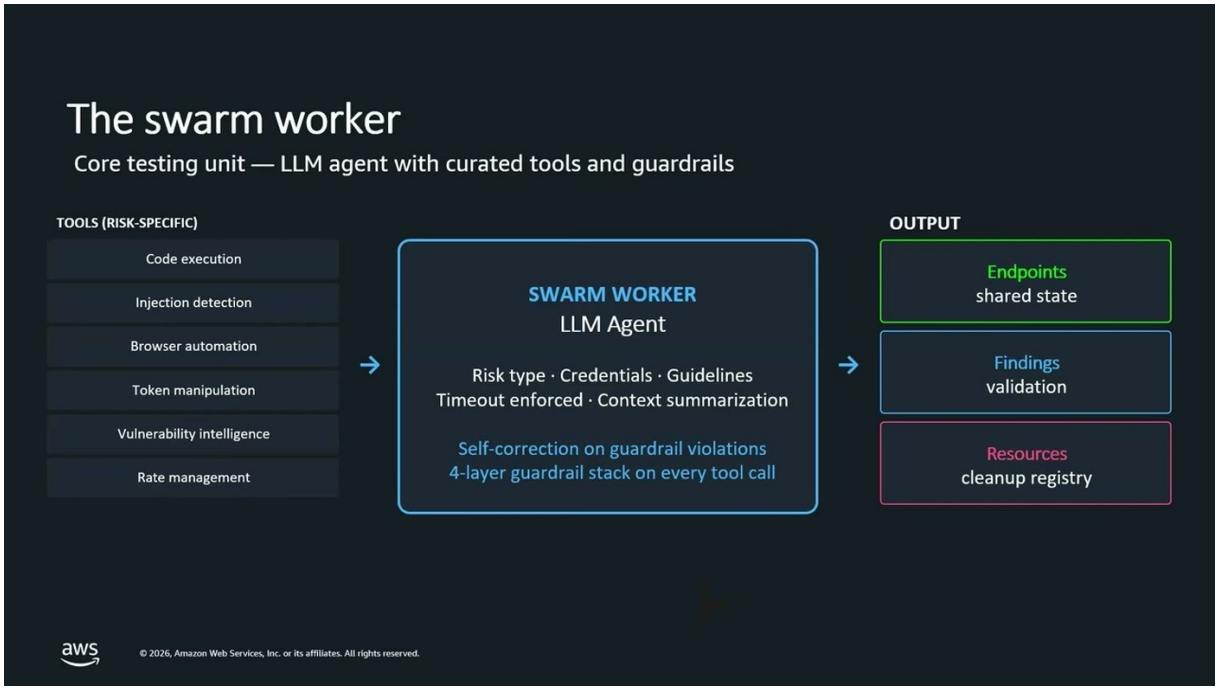
בשלב הסריקה רצים סורקים מרובים — חלקם במקביל וחלקם בשרשרת תלויות: סורק שמבצע ניתוח סטטי של הקוד מזין את מעבד המסמכים, שמזין את ה-crawlers שיוצאים לסרוק בפועל. התוצר של כל סורק נכנס ישירות ל-shared state ומשפר את הקלט לשלבים הבאים.

בשלב ההרצה (Managed Execution) מופעלים playbooks לכל סוגי הסיכונים הנפוצים באפליקציות ווב — הסלמת הרשאות, גישה לאובייקטים ללא אישור (IDOR), הזרקות SQL וכדומה. כל סיכון כזה הוא playbook אחד או יותר.

> **[7:44]** — וכל פלייבוק הזה זה בעצם מתודולוגיה איך בודקים פיילודים שמשתמשים בהם מראש אסטרטגיות מעקף של הגנות קיימות

5. יחידת העבודה: ה-Swarm Worker

כמות המשימות היא מכפלה של מספר ה-playbooks במספר ה-endpoints שנתגלו. כל משימה בודדת מורצת על ידי worker — סוכן LLM עם סט כלים שנבחר לפי סוג הסיכון שהוא אמור לבדוק.



הכלים שעומדים ל-worker, מה הוא מקבל כקונטקסט, ושלושת סוגי הפלט שהוא מחזיר ל-state

הדוגמה שניתנה: כלי שבודק חולשה בהזדהות ישתמש כנראה ב-token manipulation וב-vulnerability intelligence כדי לאתר חולשות מוכרות בחלק הספציפי בקוד שמטפל בהזדהות. המרצה הסביר למה זה חשוב מבחינת הלקוח.

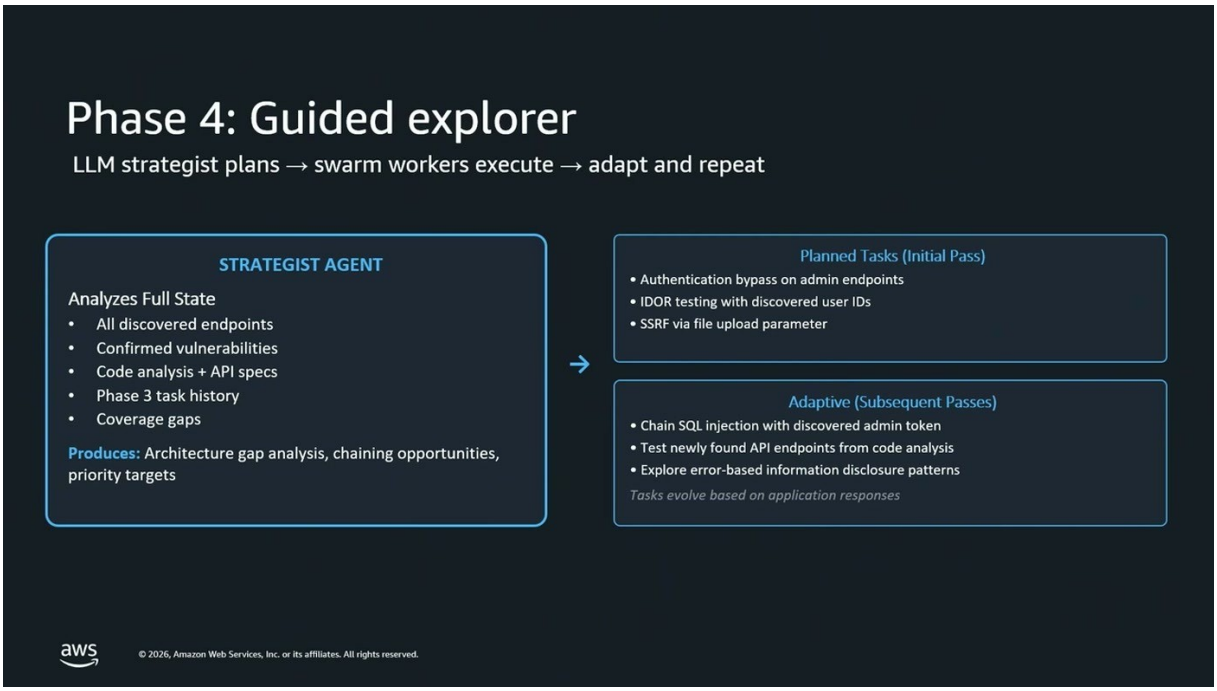
> **[7:44]** — כי זו הארכיטקטורה מודולרית היא מאפשרת לנו לשירות לקודד את המומחיות שלנו בכל אחד מהטולים האלה ומאפשרת לכם כמגנים פשוט לצרוך את המומחיות הזאת שהיא מתעתקנת מדי יום

אותה מודולריות מאפשרת גם להזין את המומחיות לגבי חולשות חדשות שמתגלות — בהנחה, שצוינה במפורש, שקצב גילוי החולשות יעלה בעידן ה-AI.

6. השלב שמייצר את ההבדל: Guided Explorer

השלב הרביעי הוא זה שמצדיק את המילה "אוטונומי". סוכן נוסף, שאחראי על אסטרטגיה, מנתח את כל מה שידוע עד כה — ה-endpoints שנסרקו, החולשות שאושרו, ניתוח הקוד וה-API specs, היסטוריית המשימות ופערי הכיסוי.

> **[9:16]** — שהאג'נט הזה יפעיל ריזנינג מבוסס AI, וייצר אנליזה על הפערים הקיימים, ייצר הזדמנויות לשרשר פגיעויות נוספות, ובעצם יסמן טרגטים שהם בעדיפות עליונה



מה הסוכן האסטרטג מנתח, ושתי דוגמאות למשימות — המעבר הראשוני מול המעברים המסתגלים שנגזרים מתגובות האפליקציה

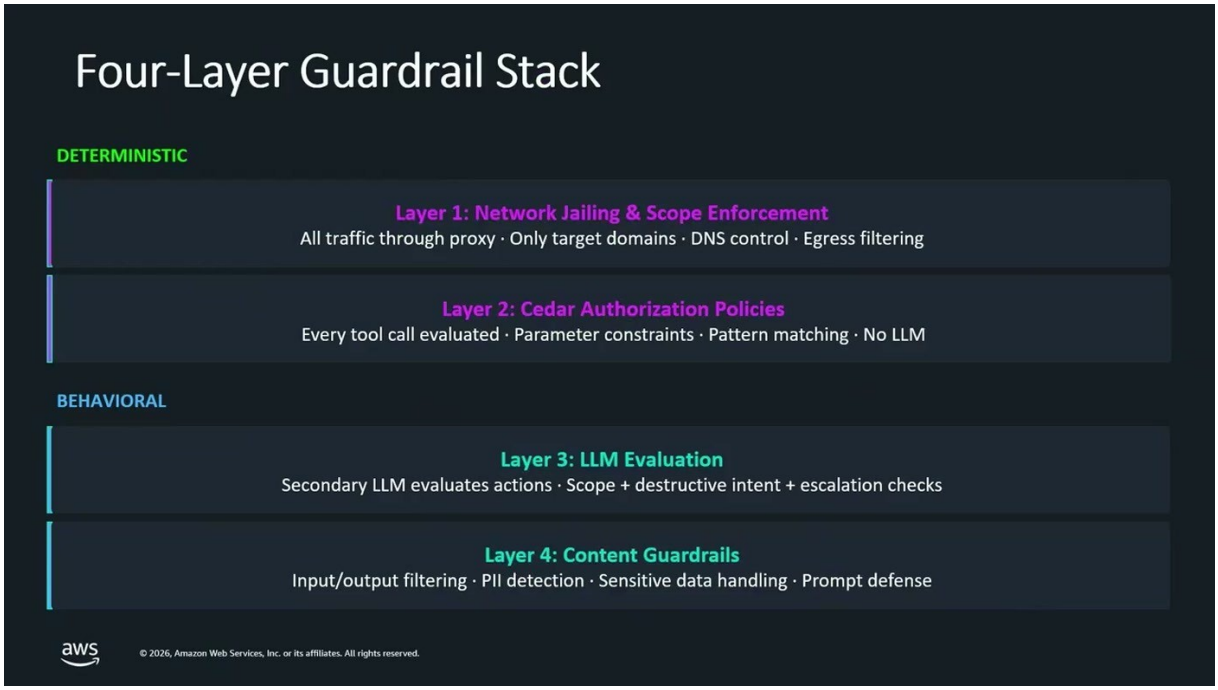
הדוגמאות שהוצגו: ניסיון להגיע לאובייקטים ללא אישור על admin endpoints שזוהו בסריקה, או השגת SSRF דרך פרמטר של העלאת קובץ. התוצרים של המשימות האלה מולידים מעבר (pass) נוסף, ואותו לופ חוזר על עצמו.

— [10:46] זה בעצם הריורד העיקרי של האג'נט הזה לייצר אימפקט על האפליקציה שאתם כמגנים יכולים להצביע עליו ולתקן

השלב החמישי הוא ניקוי: כל המשאבים והארטיפקטים שנוצרו במהלך הריצה תועדו ב-shared state לאורך הדרך, ובשלב זה הם מוסרים.

7. בטיחות: ארבע שכבות בקרה

זה היה החלק שהמרצה הקדים ואמר שרבים בקהל כנראה חשבו עליו — סוכן עם יכולות הרסניות בהגדרה, שרץ על התשתית שלכם. התשובה שהוצגה היא סטאק בן ארבע שכבות, שתיים דטרמיניסטיות ושתיים התנהגותיות.



השכבות מלמטה למעלה: בידוד רשת, מדיניות Cedar, הערכת LLM משנית, ו-guardrails על תוכן

בשכבה הראשונה, בקרות רשת מבטיחות שכל התעבורה בריצה יוצאת דרך פרוקסי ורק לדומיינים שאושרו והוכחה עליהם בעלות. בשכבה השנייה נכנסת Cedar — שפת האוטוריזציה שפיתחה AWS, שמקבלת החלטות בשיטת automated reasoning.

— [12:19] ובעצם מה שאנחנו עושים עם סידר זה אנחנו מאשררים כל טול קול שהאג'נט עושה

האישור אינו רק על עצם הקריאה אלא גם על הפרמטרים שלה, ובנוסף מורץ pattern matching על התגובה כדי לוודא שהיא תואמת למצופה.

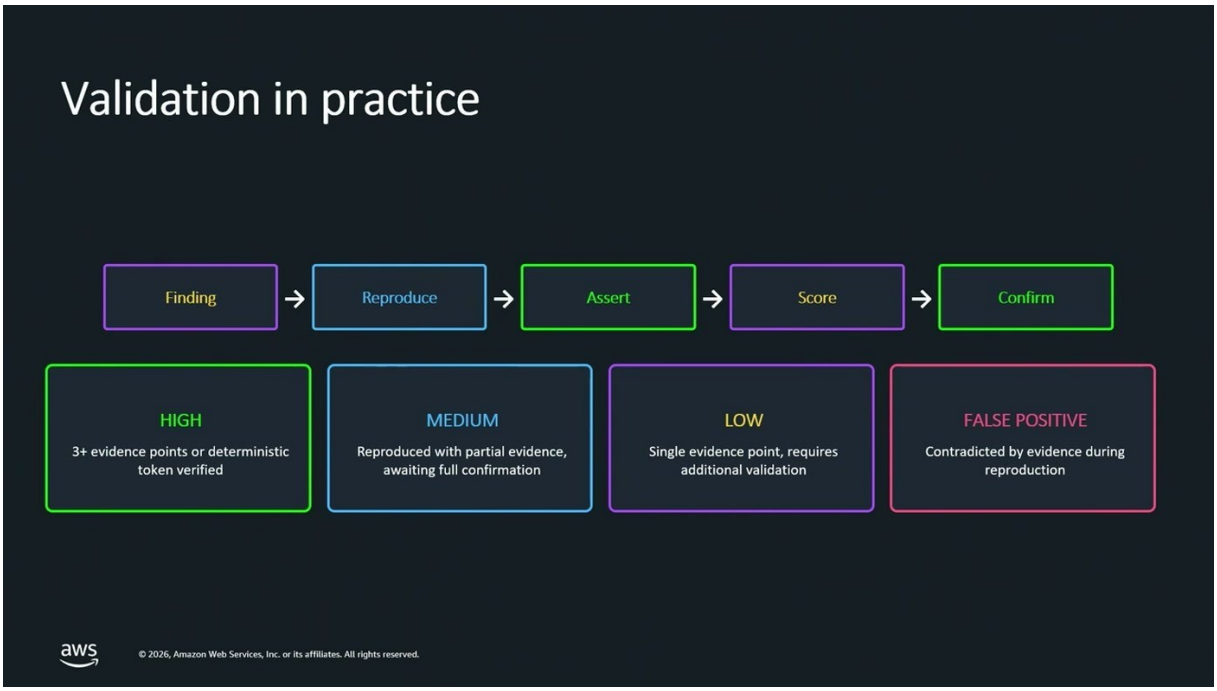
— [12:19] זה אומר, הנגנון דטרמיניסטי אין פה שום אל אלם עדיין שמחליט את ההחלטה

מעל אלה יושבות שתי שכבות מבוססות LLM: סוכן evaluation שמפעיל LLM as a judge על פעילות הכלים, ו-guardrails שמבצעים content filtering ומניעת prompt injections שעלולים לצוץ במהלך הריצה.

8. ולידציה: להבדיל בין רעש לבעיה

— [13:50] ולידציה זה בעצם היכולת להבדיל בין רעש לבעיות אמיתיות

העיקרון שהוצג: היכן שאפשר — ולידציה דטרמיניסטית. ב-TLS אפשר לאמת קריפטוגרפית שרשראות של certificates; בדפדפן אפשר לנטר את ה-DOM. היכן שאי אפשר, השירות משתמש ב-assertions — בדומה לטסטים בהנדסת תוכנה. לכל ממצא מוגדרות מראש N assertions שקובעות איך נראה ממצא מסוג זה ומה התנאים שהופכים אותו לממצא ברמת סבירות מסוימת.



ה-pipeline מזיהוי ועד אישור, וארבע רמות הביטחון עם הקריטריון לכל אחת

בפועל, כל ממצא מועבר לסוכן ולידטור נפרד שמנסה לשחזר אותו — או להפריך אותו.

[15:25] — פיינדינג שאותו אייג'נט הצליח להפריך הם בעצם פולס פוזיטיב

בסיכום החלק הראשון נוסחה ההבחנה מול הכלים הקיימים: SAST, DAST ו-SCA עובדים לרוב בבידוד, כל אחד לעצמו.

[15:25] — security agent הוא pipeline אחד שמממש גם שריקות דינמיות, גם שריקות סטטיות אבל גם את ה-exploration מבוסס AI הזה שראינו קודם

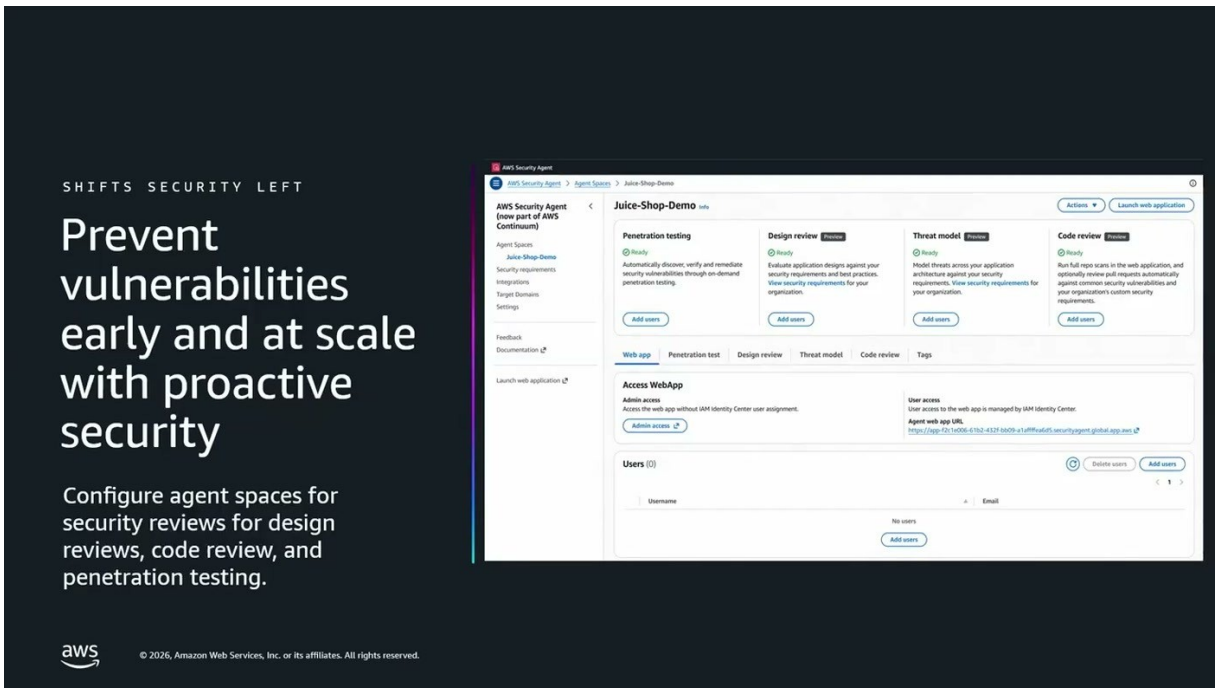
9. חוויית המשתמש: Agent Spaces ודרישות אבטחה

מאי פרץ מולא פתחה את החלק השני בשאלה לקהל, שהיא גם התזה של הסשן כולו.

— [16:58] כשמתחילים לבנות אפליקציה חדשה מתי בעצם מתחילים לחשוב על הבטחה? האם זה בזמן כתיבת הקוד או לפני עלייה לפרודקשן? התשובה היא הרבה לפני

יחידת הארגון בשירות היא Agent Space. הוא מכיל את הקונפיגורציה הרלוונטית לאפליקציה — הרשאות גישה, חיבור לריפוזיטורי — ומאגד את ארבעת המודולים.

— [16:58] אתם יכולים לחשוב על Space כעל קונטיינר לוגי לאפליקציה שלכם. לכל אפליקציה, Space משלה.

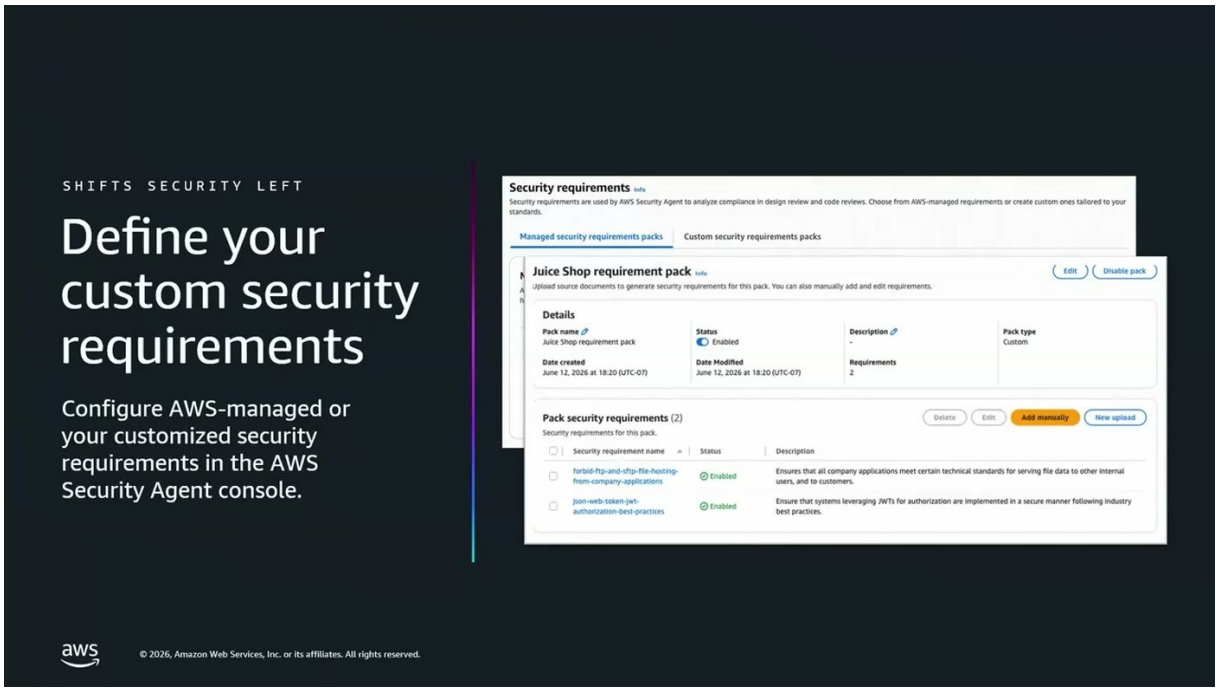


מסך ה-Agent Space בקונסול: ארבעת המודולים במצב Ready, וניהול גישת המשתמשים ל-web app

נקודה תפעולית שצוינה: אדמיניסטרטור מגדיר את ה-Space ואת ההרשאות, אך משתמשי הקצה ניגשים לאפליקציית ה-Web של AWS Security Agent — בלי גישה לקונסול של AWS. ההפרדה ל-Spaces מאפשרת לכל צוות להתמקד באפליקציה שלו.

השלב הבא הוא הגדרת מה נחשב "מאובטח" בארגון, באמצעות Security Requirements. אפשר להשתמש בסט המנוהל של AWS שממפה את ה-best practices שלה, ליצור דרישות מותאמות למדיניות הארגון, או לשלב בין השניים. הדרישות נבדקות אוטומטית בכל Code Review ו-Design Review.

— [18:31] כתיבת הדרישות נעשית בשפה טבעית כך שאין צורך לכתוב חוקים או סקריפטים

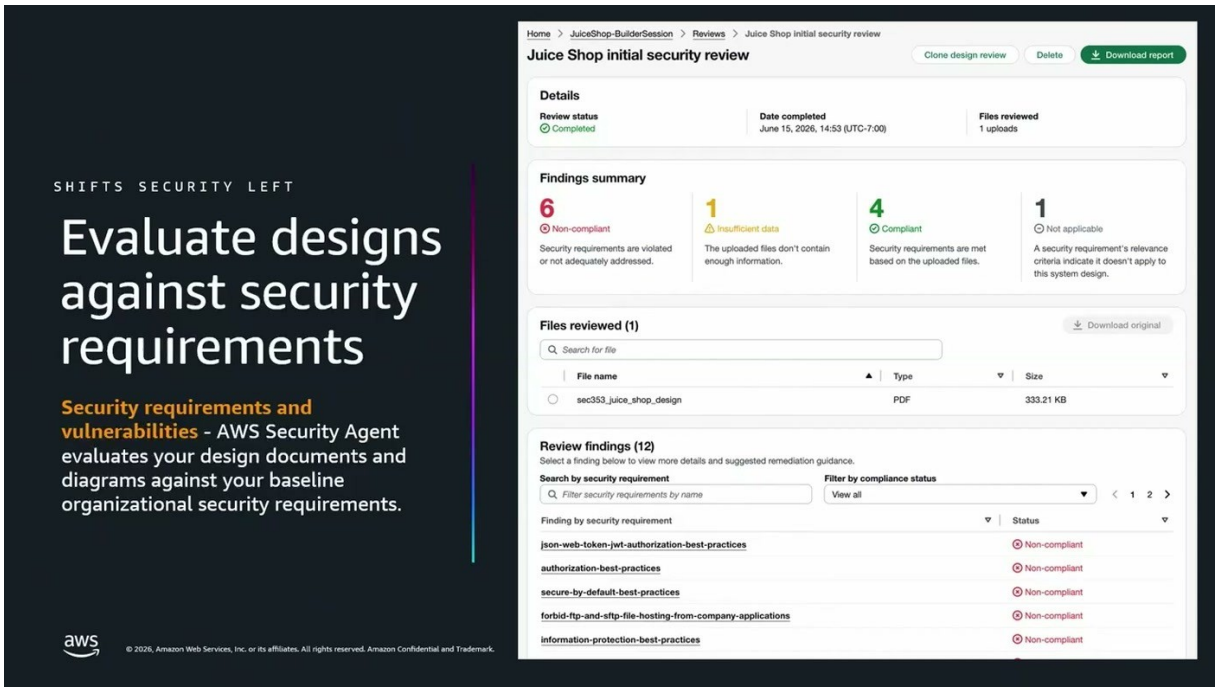


ניהול חבילות דרישות בקונסול — חבילה מנוהלת לצד חבילה מותאמת, עם שתי דרישות מוגדרות

הדוגמה שנותחה על הבמה היא דרישת JWT. לכל דרישה שני חלקים: applicability — על מה היא חלה (במקרה זה כל אפליקציית ווב או API שמשתמשת ב-JWT); ו-compliance criteria — מה ייחשב לעמידה בה. כאן: שהאפליקציה מאמתת כל טוקן שהיא מקבלת, ובודקת את אלגוריתם החתימה, את ה-issuer ואת ה-audience.

10. Threat Modeling ו-Design Review

צוואר הבקבוק שתואר: צוותי אבטחה נדרשים לאשר כל פיצ'ר או יכולת שעולה לפרודקשן. ה-Design Review נועד להקדים את הזיהוי. הסוכן סורק את מסמכי העיצוב מול דרישות האבטחה שהוגדרו ומזהה פערים ועמימויות.



דוח Design Review על OWASP Juice Shop: ארבע קטגוריות הסיווג ו-12 ממצאים מקושרים לדרישות שהופרו

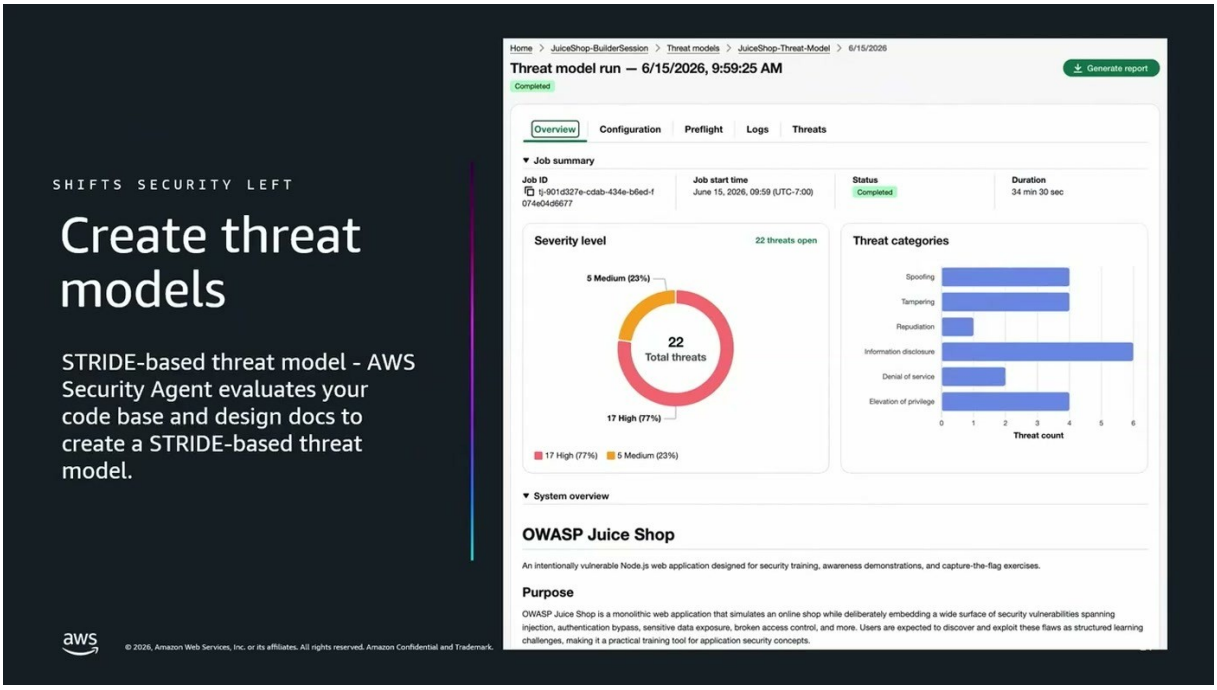
הממצאים מסווגים לארבע קטגוריות: Non-compliant, Insufficient data, Compliant ו-Not applicable. המסמכים שאפשר להזין כוללים מסמכי מוצר, דיאגרמות ארכיטקטורה ומסמכי threat modeling — הסוכן מסתכל

עליהם כמכלול אחד. בממצא שהודגם, המסמכים ציינו הקמת שרת FTP לשמירת מסמכי החברה, בניגוד לדרישה שאוסרת זאת; הדוח כלל גם את ההסבר למה וגם remediation guidance.

[21:38] — אנחנו חוזרים כאן לעולמות של פרואקטיב סקיוריטי. זיהוי פערי הבטחה עוד לפני שנרשמה שורת קוד אחת

יכולת ה-Threat Modeling לוקחת את זה צעד קדימה. היא נשענת על שני מקורות: Scope Docs (מסמכי עיצוב, דיאגרמות, API specs) וקוד המקור. הסוכן מצליב ביניהם, לומד את האפליקציה ומחפש את נקודות התורפה. התוצר מורכב מ-System Overview ומרשימת איומים פוטנציאליים.

[23:12] — לכל ממצא יש סיווג לפי סטרייד, חומרת הממצא וגם המלצות פרקטיות למיטיגציה



ריצת Threat Model שהושלמה תוך 34 דקות: 22 איומים, פילוח חומרה ופילוח לפי קטגוריות STRIDE

11. Code Review ופנטסט

בשלב הפיתוח הסוכן מלווה בשתי דרכים, לפי ה-workflow של הצוות: סריקת ריפוזיטורי מלא, אינטגרציה עם GitHub לסריקת כל Pull Request, או שילוב. האינטגרציה נעשית באישור אפליקציית ה-Security Agent ב-GitHub, ברמת הארגון או ברמת האפליקציה. לכל Agent Space מגדירים על בסיס מה ירוץ ה-review — פגיעויות נפוצות, דרישות האבטחה שהוגדרו, או שניהם.

> **[24:45]** כאשר האגנט מזהה חולשה הוא מפרסם נערה בפול ריקווסט במקום הרלוונטי המפתח נשאר בסביבת העבודה שלו בלי לעשות איזשהו contact switching

בדוגמה שהוצגה, קטע קוד השתמש ב-JWT verification עם שני אלגוריתמים קריפטוגרפיים — הפרה של הדרישה שהוגדרה קודם. נקודה שהודגשה: ב-PR רואים רק מה השתנה, ולכן חיבור הריפוזיטורי המלא נותן לסוכן קונטקסט על הארכיטקטורה, על ה-trust boundaries ועל ה-data flows — ומאפשר לאתר פערים שנוצרים דווקא מהאינטראקציה בין החלקים.

הפנטסט הוא, בלשונה, "המקום שבו הכל מתחבר".

> **[26:17]** האג'נט לא מגיע לכאן כקופסה שחורה עם URL הוא לומד את האפליקציה שלנו הוא מכיר את האפליקציה שלנו מהקונטקס שנתנו לו קודם לכן

ההגדרה כוללת Target URL — דומיין ציבורי או endpoint ב-VPC פרטי; בחירת הבדיקות, כאשר ברירת המחדל היא OWASP Top 10 לכל אפליקציית ווב או API; אפשרות להחריג בדיקות שאינן רלוונטיות כדי לקצר את זמן הריצה; החרגת כתובות מסוימות; ו-custom HTTP header שמתייג את התעבורה מהסוכן, כדי שה-SOC הארגוני יזהה אותה כפעילות לגיטימית ויאפשר אותה בכלי ההגנה.

> **[27:49]** הוא בונה את תרחישי התקיפה בהתאם לאפליקציה ומקצר את התהליך משבועות לימים

דוח הפנטסט מציג לכל ממצא את רמת הביטחון של הסוכן ואת חומרת הממצא, את תיאור הפגיעות ומיקומה בקוד, את תרחיש הניצול, steps to reproduce המבוססים על הקוד עצמו, ובשלב הרמדיאציה — קטע קוד לדוגמה לתיקון. בדוגמה שהוצגה הסוכן זיהה שאפשר לעקוף את מנגנון ה-JWT verification באמצעות טוקן שהמשתמש מייצר לעצמו, ולהתחזות למשתמש אחר עד לשליטה רחבה על האפליקציה.

12. AWS Continuum

החלק האחרון היה ההכרזה: AWS Continuum, שירות האבטחה שלתוכו נכנס Security Agent.

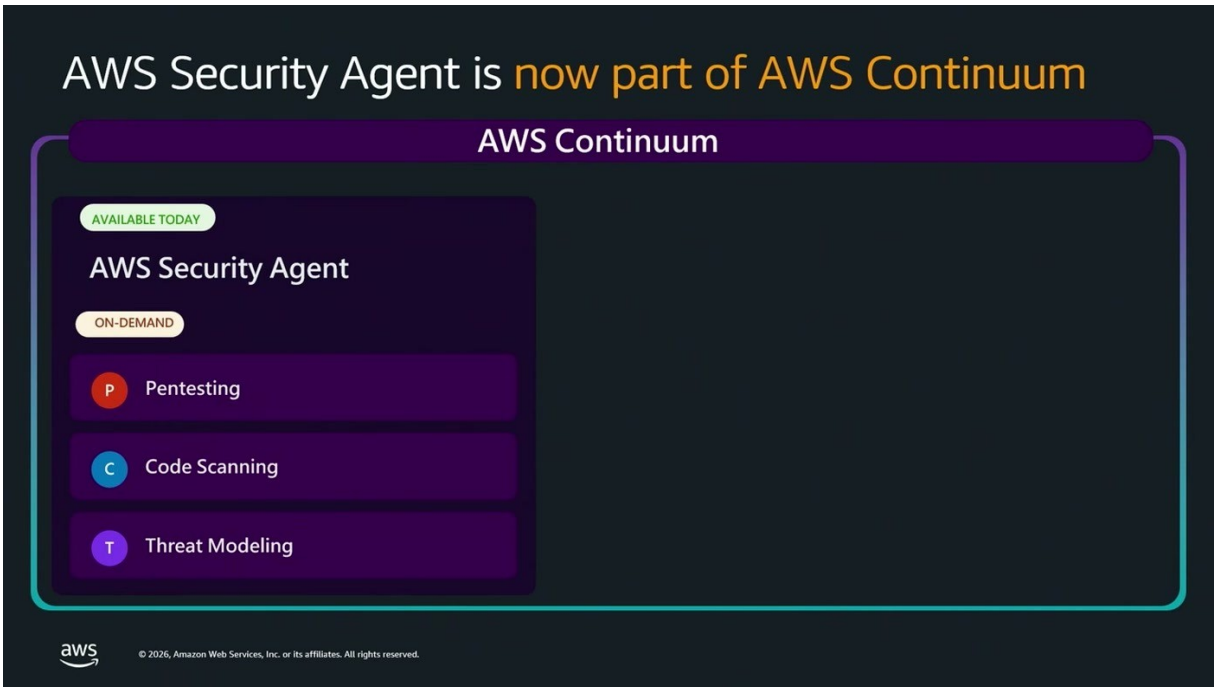
> **[29:24]** שירות ההבטחה של AWS, שנועד לסייע ללקוחות לעשות security at machine speed

שלושת הכוחות שהוצגו כרקע: תוקפים מנצלים יכולות אג'נטיות כדי לזהות ולנצל פגיעויות מהר מאי פעם; מפתחים משתמשים בכלי AI ומגדילים את ה-code base ואיתו את שטח התקיפה; וצוותי האבטחה עדיין עובדים במודל ריאקטיבי — הכלים מזהים בעיות אחרי שהן כבר רצות בפרודקשן.

הלופ של Continuum מורכב מארבעה שלבים סביב Business Impact: Detect, Prioritize, Validate, Remediate. בשלב הזיהוי, Continuum לומד את הפגיעויות שצוותי האבטחה התמודדו איתן בעבר ומריץ סריקה לזיהוי נוספות, לצד סורקים אחרים בסביבה כמו מוצרי Cloud Security. החידוש המרכזי הוא בתיעדוף: במקום להסתמך רק על הקונטקסט של הממצא (למשל האם הרכיב נגיש לעולם או מחובר לדאטה רגיש), Continuum מוודא שהממצא אכן ניתן לניצול.

> **[30:57]** קונטיניום יכול להריץ שחזור של התקיפה בסביבת סנדבוקס מבודדת ולתת לנו תובנות על אם הממצא אכן אקספלוידיבל בסביבה שלנו

בשלב הרמדיאציה, לפי התיאור שניתן, Continuum פועל כחבר צוות: משנה IaC, מתקן Security Groups או IAM Policies לפי הממצא. הלופ רץ באופן מתמשך ומעביר את האבטחה מרמת האפליקציה לרמת התשתית.



מיצוב השירותים: Security Agent זמין היום ורץ on-demand, בתוך המסגרת הרחבה של Continuum

חלוקת הזמנים שנמסרה: Security Agent הוכרז ב-re:Invent וזמין היום, עם יכולות שרצות on-demand. ביוני הוכרז Continuum כאבולוציה שלו, שמשנה את אופן הפעולה מ-on-demand לפעולה מתמשכת.

— [32:32] Continuum Code Vulnerabilities נמצא כיום בגייטד פריביו ומלווה את מחזור חיי הפעולה משלב הזיהוי ועד לשלב הפעולה

נקודה מעשית לא נדרשת מיגרציה בין השירותים. בלשון המרצה: "השירות של Continuum מתפתח לתוך עצמו כך שלא תצטרכו לבצע איזושהי מיגרסיה בהמשך אותו דאטה, אותו קונסול, ההשקעה שלכם נשמרת" [32:32].

13. מה לוקחים מכאן

> **— [34:06]** זה ההבנה שהבטחה יכולה להיות פרואקטיבית ולעשות הרבה לפני שהגענו לפרודקשן

- **יש נתיב התנסות ללא עלות מיידית.** בסיום הוצע free trial של חודשיים לכל חשבון AWS [34:06] — מספיק כדי להריץ Design Review ו-Threat Model על אפליקציה אחת ולראות מה חוזר.
- **הערך הראשוני הוא דווקא בשלב העיצוב.** Design Review ו-Threat Modeling רצים על מסמכים בלבד, לא דורשים endpoint חי ולא נגיעה בפרודקשן — זו נקודת הכניסה בעלת הסיכון הנמוך ביותר.
- **ההשקעה האמיתית היא בדרישות האבטחה.** הדרישות נכתבות בשפה טבעית, פעם אחת, וחלות על כל האפליקציות. זה הרכיב שהופך את השירות מסורק גנרי לכלי שמאכף את המדיניות של הארגון הספציפי.
- **לפני הרצת פנטסט יש עבודת תיאום פנימית.** הוכחת בעלות על הדומיין, החרגת כתובות רגישות, ותיוג התעבורה ב-custom HTTP header כדי שה-SOC לא יתייחס לריצה כתקיפה.
- **שאלות פתוחות שהסשן לא ענה עליהן.** לא הוצגו מחירים, לא הוצגו נתוני דיוק או שיעורי false positive מדידים, ולא נמסרו זמינות אזורית או מגבלות רגולטוריות. אלה נושאים לבירור מול צוות החשבון.

נספח: מילון מונחים

מונח	מה זה	איפה הופיע
Agent Space	קונטיינר לוגי לאפליקציה אחת — הרשאות, חיבור לריפוזיטורי וארבעת המודולים	[16:58]
Security Requirements	דרישות אבטחה בשפה טבעית, מנוהלות או מותאמות, שנבדקות בכל review	[18:31]
PentestState	ה-state המשותף שכל שלב מעשיר: endpoints, סיכונים, credentials, מסמכים, כיסוי ומשאבים	[6:12]
Playbook	מתודולוגיית בדיקה לסוג סיכון אחד: payloads ואסטרטגיות מעקף הגנות	[7:44]
Swarm Worker	סוכן LLM שמריץ משימה בודדת עם סט כלים שנבחר לפי סוג הסיכון	[7:44]
Guided Explorer	השלב שבו סוכן אסטרטג מנתח את כל ה-state ומתכנן שרשור פגיעויות	[9:16]
Cedar	שפת אוטוריזציה של AWS, משמשת לאישור דטרמיניסטי של כל קריאת כלי	[12:19]
Assertions	תנאים שהוגדרו מראש לכל סוג ממצא, שקובעים את רמת הביטחון שלו	[15:25]
STRIDE	מתודולוגיית סיווג איומים שלפיה מסווגים ממצאי ה-Threat Model	[23:12]
AWS Continuum	השירות שלתוכו נכנס Security Agent, עם לופ מתמשך של Detect–Prioritize –Validate–Remediate	[29:24]