לבנות סוכני AI מאובטחים-בתכנון, בסקייל

TLV-SEC301 — סיכום סשן, AWS Summit Tel Aviv 2026

Sefi Avrech, Sr. Solutions Architects, AWS | Melih Bahar, AI/ML & Engineering Lead,-ו Shlomy Cohen Dream

10 בספטמבר 2026 | משך: 41 דקות | הופק מהקלטת הסשן

סיכום מאת קובי אלמוג

על המסמך הזה

המסמך מסכם את הסשן TLV-SEC301 מתוך AWS Summit Tel Aviv 2026 (תראק ,AI Cloud Ops Infrastructure & Security, רמה 300), בהתבסס על ההקלטה המלאה. הוא נבנה מתמלול אוטומטי של ההקלטה ומהסליידים שחולצו מהווידאו עצמו, כך שאפשר לחזור לתוכן בלי לצפות שוב בארבעים הדקות. חותמות הזמן בגוף המסמך מפנות לנקודה המדויקת בהקלטה.

איך לקרוא את זה

- **פרק 1 — תקציר מנהלים:** חמש דקות קריאה. מה נטען בסשן ומה רלוונטי למי שבונה סוכנים בארגון.
- **פרקים 2–5 — התיאוריה של האיום:** למה אמצעי ההגנה המסורתיים לא מספיקים, מה בדיוק שונה במודל, ואיך נראה משטח התקיפה.
- **פרקים 6–10 — Threat modeling הלכה למעשה:** ארבע השאלות, יוזקייס פיננסי אחד, ארבע מתקפות והמיטיגציות שהוצעו להן ב-AWS.
- **פרק 11 — Dream מהשטח:** שתי דוגמאות פרודקשן אמיתיות מתוך מערכת multi-agent, והפריימוורק הפנימי שבנו סביבן.
- **פרק 12 — מה לוקחים מכאן, ואחריו מילון מונחים.**

הערת מקור המסמך הופק מתמלול אוטומטי של ההקלטה. התמלול מדויק בקטעים המקצועיים, אך מונחים לועזיים ושמות מוצרים הופיעו בו בשיבוש פונטי (למשל "אג'נט קור" = AgentCore) ותוקנו ידנית מול הסליידים. הציטוטים אומתו מול ההקלטה בחותמת הזמן המצוינת ומובאים כלשונם, כולל שיבושי התמלול. שמות הדוברים ותפקידיהם נלקחו משקופית הפתיחה, לא מהתמלול.

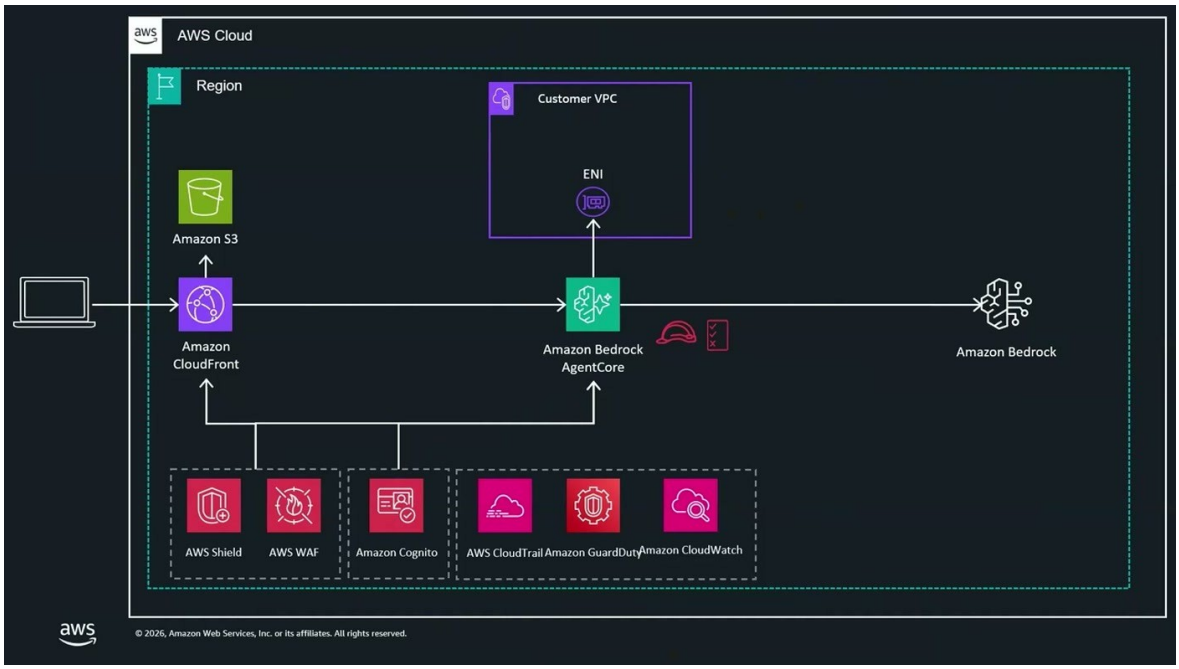
שקופית הפתיחה: שני ארכיטקטים מ-AWS ולקוח — Dream — שמביא את הצד של הפרודקשן

1. תקציר מנהלים

הסשן מיפה שלושה מקומות שבהם סוכנים פוגשים אבטחה — בניית סוכנים מאובטחים, הגנה מפני סוכנים עוינים, ושימוש בסוכנים לצורכי הגנה — ובחר להתמקד בראשון בלבד: "אז אג'נטים פוגשים סקיוריטי בשלושה מקומות שונים" [0:00]. מכאן ואילך זה סשן על עיצוב, לא על זיהוי תקיפות.

- **הבעיה היא ארכיטקטונית, לא התנהגותית.** במודל עצמו אין זהות ואין הרשאות: "אין פה אידנטיטי, אין פה פרמישנז, אין פה אוטוריזיישן, שום דבר מאלה בארכיטקטורה של המודל" [3:04]. לכן כל בקרה חייבת לשבת מחוץ למודל — אחרת בקרה דטרמיניסטית הופכת לסטטיסטית.
- **אין הפרדה בין קוד לדאטה.** system prompt ו-user prompt מתורגמים שניהם לטוקנים ונכנסים לאותו מודל: "אין לנו באמת הפרדה ארכיטקטונית בין אינסטרקשן או קוד לבין דייטה" [4:37]. זו השורש של כל מתקפות ההזרקה.
- **אמצעי ההגנה המסורתיים נשארים — הם פשוט לא מספיקים.** הסשן נפתח בארכיטקטורה קלאסית (CloudFront, WAF, Shield, VPC endpoints) ואז אמר עליה: "תשימו לב שכל האמצעי ההבטחה שהזכרתי הם אמצעי ההבטחה מסורתיים שלא רלוונטיים במיוחד לאפליקציות אג'נטיות" [1:33].
- **המתודולוגיה שהוצעה היא threat modeling בארבע שאלות.** מתוך ה-Security Pillar של AWS Well-Architected: על מה אנחנו עובדים, מה יכול להשתבש, מה נעשה לגבי זה, והאם עשינו עבודה מספיק טובה [12:11]. ה-OWASP Top 10 for Agentic Applications 2026 שימש כקטלוג האיומים.
- **המיטיגציות שהודגמו הן מוצרי AWS קונקרטיים.** Amazon Bedrock Guardrails לפני המודל, ואז AgentCore Identity (2LO מול 3LO), AgentCore Memory עם namespaces, ו-AgentCore Gateway כנקודת האכיפה לכלים.
- **הצד של הלקוח הראה שהמיטיגציה האמיתית היא דטרמיניזם, לא עוד LLM.** Dream מריצה classifier על גרף זיכרון במקום LLM-as-a-judge, ומסכמת: "memory is evidence והוא לא authority" [33:38].

2. נקודת המוצא: אפליקציה אג'נטית היא קודם כל אפליקציה



הארכיטקטורה שנבנתה על הבמה שכבה-שכבה: CloudFront, ALB, VPC endpoint אל Bedrock, ומסביב, WAF, Shield, Cognito, GuardDuty ו-CloudTrail

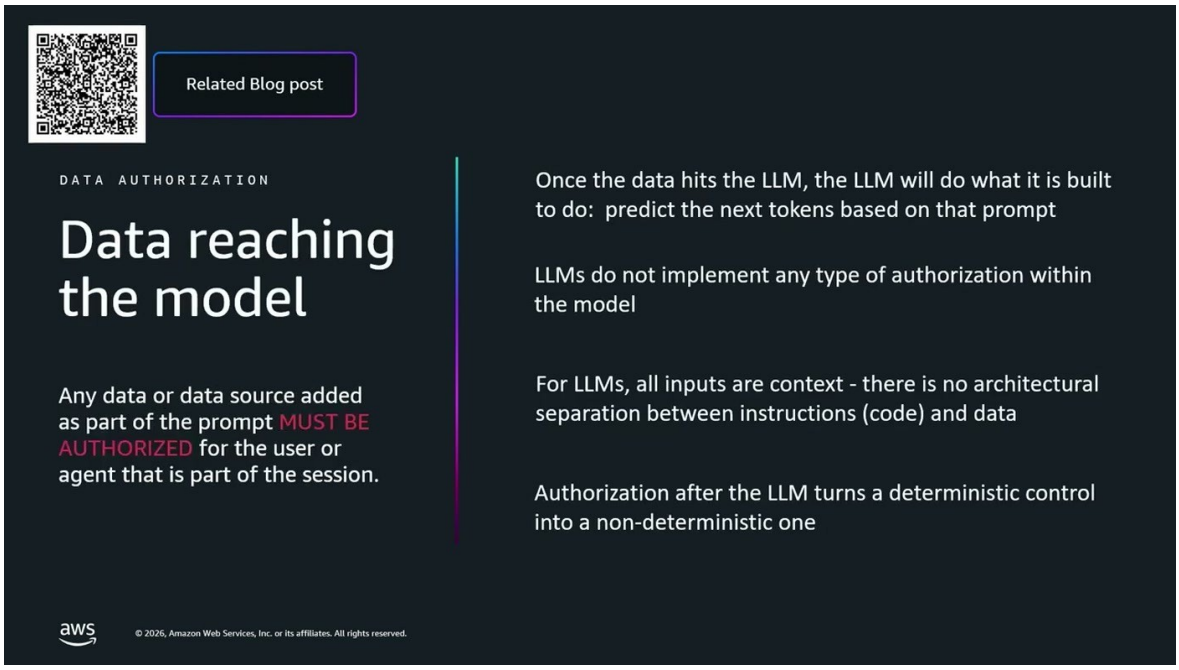
הסשן נפתח דווקא במה שלא חדש. ארכיטקט AWS בנה על המסך אפליקציה אג'נטית סטנדרטית: מודל דרך Amazon Bedrock, פנייה אליו דרך VPC endpoint "על מנת לצמצם, למנוע את החשיפה לאינטרנט", Application Load Balancer שעושה offloading של הגנת שכבות 6 ומטה אל AWS — "זה שימוש מצוין בשרד רספונסיביליטי מודל" — ו-CloudFront שמרחיק עוד יותר את משטח התקיפה [1:33]. מעל זה security groups, roles, permissions, WAF, DDoS protection, זהויות, observability ו-detection.

ואז הגיעה ההודאה: כל אלה הם אמצעים מסורתיים שאינם רלוונטיים במיוחד לאפליקציות אג'נטיות. אז למה לפתוח בהם? "אפליקציה אג'נטית היא קודם כל אפליקציה" ו-Defense in Depth נשארת אסטרטגיית ההגנה [3:04]. זו נקודה שחזרה גם בסיכום [39:47] — המסגרת הישנה לא מוחלפת, רק לא מספיקה.

3. מה באמת השתנה: המודל כמנוע סטטיסטי

כדי להסביר מה השתנה, הסשן הרים את מכסה המנוע של ה-transformer: הקלט מתורגם לטוקנים, עובר שכבות שחוזות את הטוקן הבא הסביר ביותר סטטיסטית, והטוקן מצטרף לקלט וחוזר חלילה עד טוקן סיום. "זה מודל שבסופו של דבר הוא מודל סטטיסטי" [3:04].

מכאן שתי מסקנות שחוזרות לאורך כל הסשן. הראשונה: בארכיטקטורת המודל אין זהות, אין הרשאות ואין authorization — ולכן "כל אמצעי הבטחה, פרמישנס, כל דבר שמתחשב באיידנטיטי, נצטרך לעשות מחוץ למודל" [4:37]. השנייה נובעת מה-Converse API של Bedrock, שהוצג על המסך: יש user prompt ויש system prompt, והם נראים כמו הפרדה בין הוראות לדאטה — אבל "בסופו של יום הסיסטם פרומט והיוזר פרומט שניהם ביחד מתורגמים לטוקנים, למספרים" [4:37].



ארבע האמירות שמהן נגזרת כל שאר ההרצאה — ובראשן: authorization אחרי ה-LLM הופך בקרה דטרמיניסטית ללא-דטרמיניסטית

> **עיקרון** העיקרון התכנוני של הסשן, בשורה אחת: כל דאטה או מקור דאטה שנכנס לפרומט חייב להיות מורשה עבור המשתמש או הסוכן שבסשן — לפני שהוא מגיע למודל, לא אחרי.

4. משטח התקיפה: קלט, פלט וסופליי צ'יין

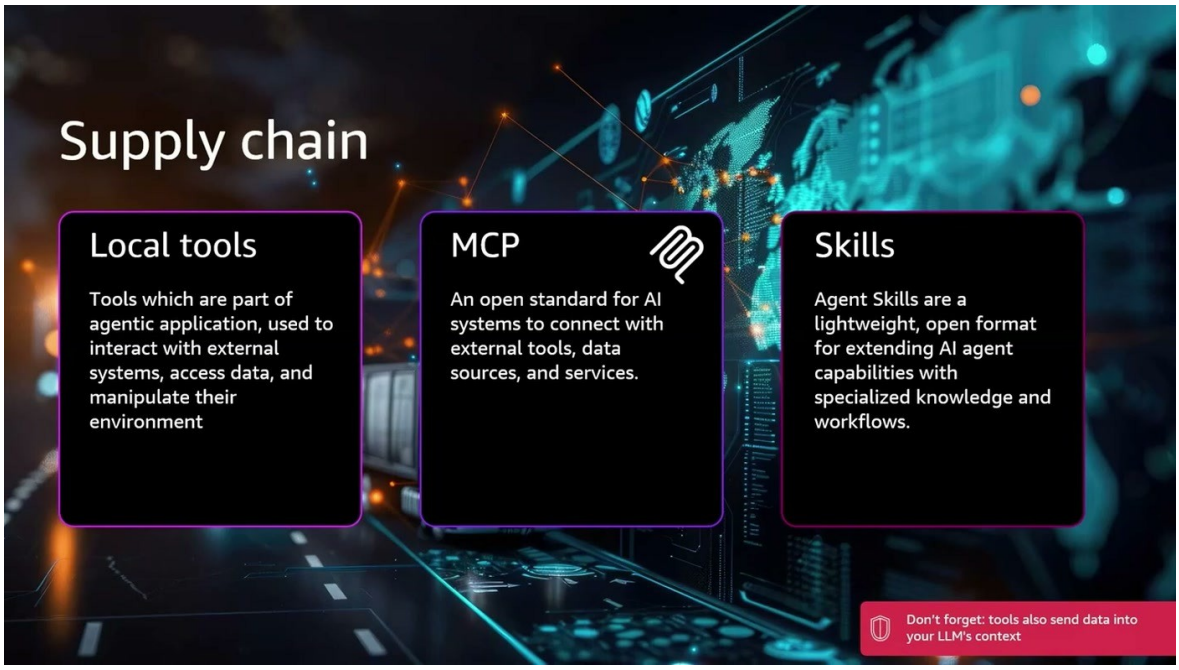
הקלטים

הפרומט של המשתמש הוא רק ההתחלה. גם משתמש פנימי יכול להעלות attachment שמגיע "אלוהים יודע מאיפה"; סוכנים שולפים מידע מ-data stores ו-knowledge bases; כלים מחזירים דאטה שנכנס לקונטקסט; והזיכרון מאינטראקציות קודמות נכנס גם הוא [6:08]. על כל אחד מהמקורות האלה צריך לשאול שתי שאלות:

> **— ארכיטקט AWS, [6:08]** שאלה ראשונה אם זה trusted האם הדאטה שמגיעה הוא trusted או האם אני פה חשוף לתוקף והשאלה השנייה זה האם כל הדאטה הזה הוא מורשה ליוזר בקונטקסט שבו הוא נמצא

הפלטים

בצד השני, התוצאה משפיעה על business outcome — וזה כשלעצמו יעד לתקיפה. כלים עם גישה למערכת הקבצים או עם יכולת להריץ קוד מקומי; כלים שמשנים state במערכת עסקית; כתיבה לזיכרון שתשפיע על ה-workflow הנוכחי ועל הבאים אחריו; ויחסי אמון בין סוכנים בסביבות multi-agent, שבהם "תוקף יכול לנצל את הטרסט הזה ולהמשיך את ההתקפה קדימה אל האג'נט הבא" [7:38].



שלושת ערוצי הסופליי צ'יין האג'נטי; בפינה התזכורת — כלים מזרימים גם דאטה אל תוך הקונטקסט של המודל

הסופליי צ'יין

- **Local tools —** כלים שמסופקים יחד עם האפליקציה; הסופליי צ'יין שלהם הוא זה של האפליקציה, ולכן מוכר.
- **MCP servers —** "שהרבה פעמים רצים מחוץ לארגון שלי" ויוצרים יחסי trust ו-dependency מול גורם חיצוני. ההצעה שנשמעה: "אולי כדאי להכניס אותם פנימה כדי לשלוט עליהם" [9:09].
- **Skills —** "פאטרן היום מאוד מאוד נפוץ", שצריך להתייחס אליו כאל קוד שנכנס אל תוך הפרומט, ולא מעט פעמים גם כסקריפטים שרצים מקומית [9:09].

ומעל הכול — אוטונומיה

ההבדל המהותי: "אגנט זה יצור אוטונומי או לפחות אנחנו רוצים שהוא ייצור אוטונומי" — הוא מקבל מטרה ומתכנן בעצמו את ה-workflow, הצעדים והכלים. "זה גם כלי חזק מאוד כשהוא בידיים של תוקף" [9:09]. נלוות לכך שתי תופעות: ריבוי נקודות אינטגרציה ("friction points פה יש לנו פשוט by design לא מעט כאלה") והתנהגות שמתפתחת עם הזמן — כלומר גם תוקף יכול להשפיע על מה שהסוכן לומד [10:39].

5. המסגרת: OWASP Top 10 for Agentic Applications

עשרת הסיכונים כפי שהוצגו על המסך; בפינה נרמז גם ה-OWASP Agentic Skills Top 10 הנפרד

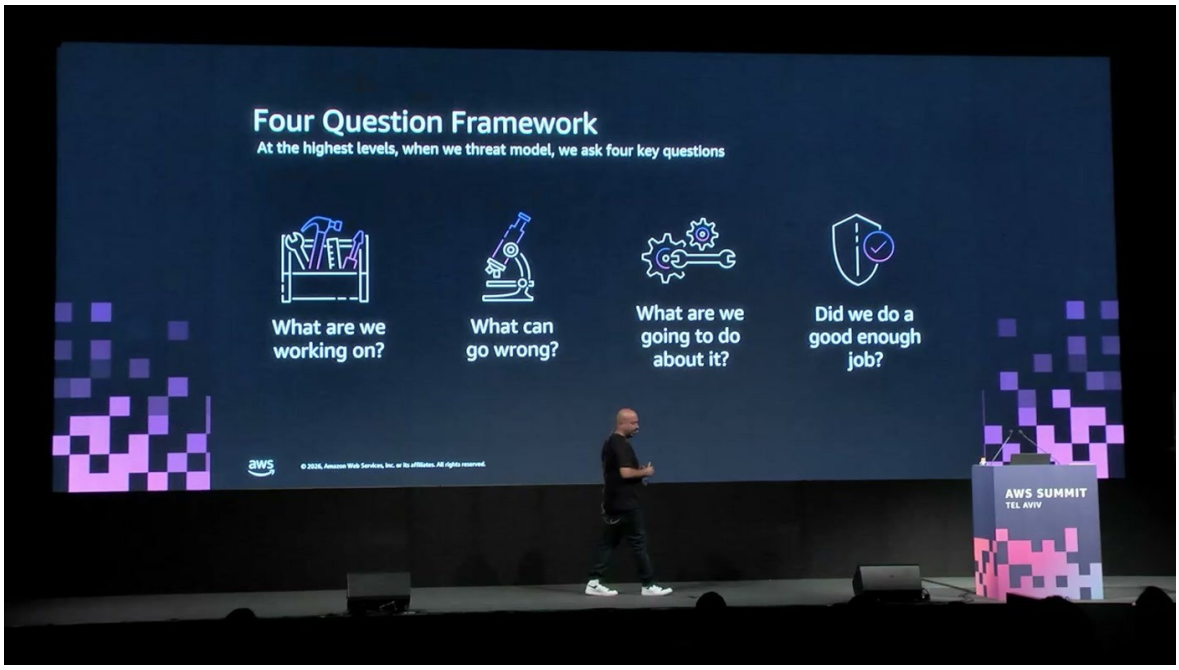
אחרי מיפוי האיומים, הסשן פנה לקטלוג מסודר. ה-Top 10 לאפליקציות אג'נטיות הוזכר כמסמך חדש ("נדמה לי, יצא ביולי השנה"), והמגיש היה כן לגבי תפקידו: "שום דבר כאן כמעט לא אמור להיות חדש לכם זה פשוט לעשות לזה פריימינג" [10:39]. מתוך העשרה נבחרו ארבעה שילוו את שאר הסשן.

מזהה	סיכון	מה הוא במילים פשוטות
ASI01	Agent Goal Hijack	התוקף מזריק הנחיה זדונית ומשנה את מטרת הסוכן — ומכאן הסוכן עובד עבורו
ASI03	Identity and Privilege Abuse	ניצול מנגנוני האמון וההרשאות של הסוכן כדי לשלוף נתונים שלא נועדו לו (confused deputy)
ASI06	Memory & Context Poisoning	הזרקת מידע שגוי לזיכרון ארוך-טווח, כך שגם בשיחות הבאות התשובות יישארו מורעלות
ASI02	Tool Misuse and Exploitation	הסוכן מבצע פעולת API אמיתית — אבל עבור התוקף

> **AWS, [16:48] — ארכיטקט** עכשיו צריך להבין, הכלים עושים את מה שהם נועדו לעשות הם פשוט עושים את זה עבור אותו תוקף

6. Threat modeling בארבע שאלות

הדובר השני פתח בסקר ידיים באולם: מי מכיר threat modeling, מי משתמש בו — ואז "מי משתמש בו לאפליקציה אג'נטית? אחד, שתיים" [12:11]. זו הנחת המוצא לחלק המעשי.



ארבע השאלות של Well-Architected Security Pillar, שמכתיבות את מבנה שאר הסשן

> **— ארכיטקט AWS, [12:11]** שאלה ראשונה על מה אנחנו עובדים, שאלה שנייה מה יכול להשתבש, שאלה שלישית מה אנחנו הולכים לעשות לגבי זה ושאלה רביעית ואחרונה שהיא תמיד קריטית, האם עשינו עבודה מספיק טובה

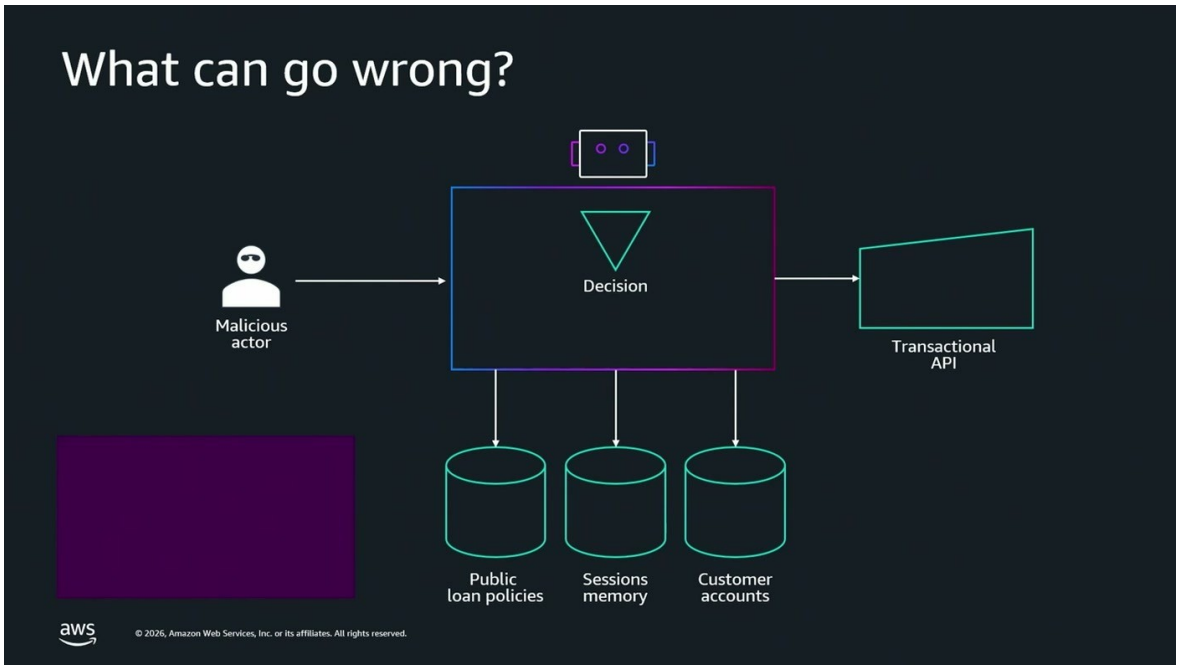
7. שאלה 1 — על מה אנחנו עובדים: סוכן שירות לקוחות במוסד פיננסי

היוזקייס שנבחר: "מוסד פיננסי כזה או אחר, שרוצה להשמיש סוכן של שירות לקוחות בתוך האתר שלו" [13:45].
לסוכן הוגדרו ארבע יכולות, בסדר עולה של סיכון:

- לענות על שאלות כלליות לגבי פוליסות שמפורסמות באתר, למשל בנושא לקיחת הלוואות
- לענות על סטטוס החשבון של המשתמש עצמו
- לקבוע האם המשתמש זכאי להלוואה — כלומר האם הוא עומד בתנאים
- לתת הלוואה בפועל: "יכולת לתת הלוואה בצורה אקטיבית ולא איזשהו אישור כזה או אחר" [13:45]

ההערה החשובה בשלב הזה הייתה מתודולוגית: בעולם האמיתי חלק מהעבודה הוא גם מיפוי ה-data flow, וזה פשוט לא נכנס לזמן הסשן.

8. שאלה 2 — מה יכול להשתבש: שרשרת של ארבע מתקפות



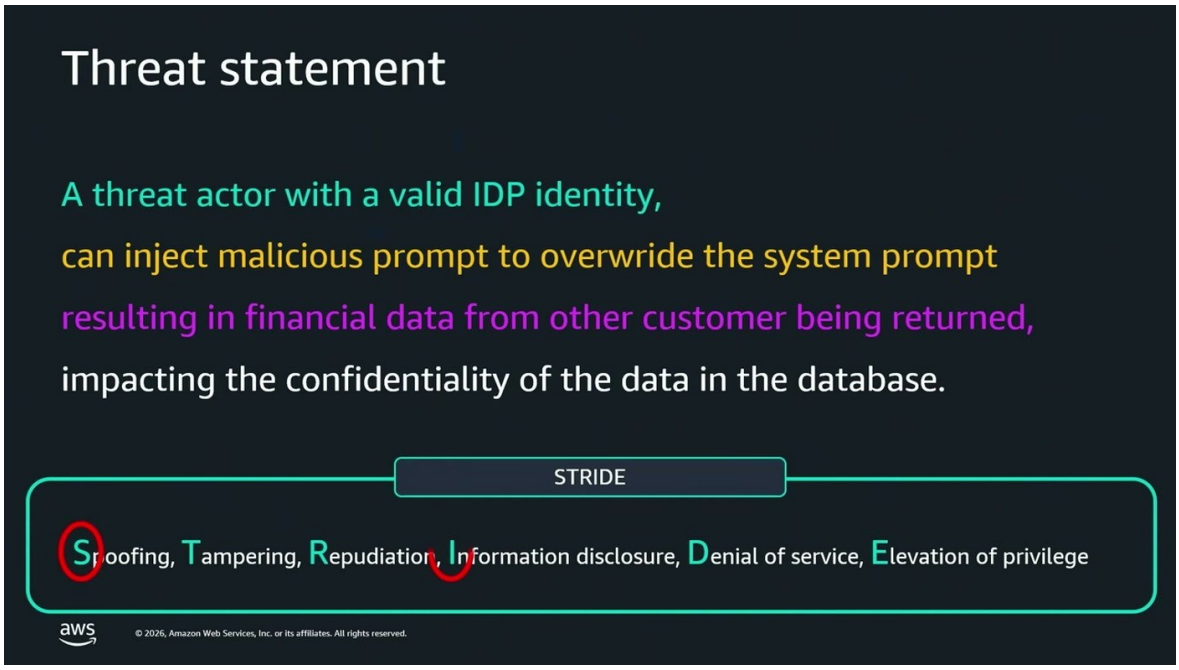
שחקן זדוני בעל הרשאות ולידיות מול סוכן ההחלטה, שלושת מאגריו (פוליסות ציבוריות, זיכרון סשנים, חשבונות לקוח) וה-Transaction API בקצה

ארבע הפגיעויות הוצגו כשרשרת אחת ולא כארבעה אירועים נפרדים. הפתיחה היא Agent Goal Hijack: התוקף מזריק הנחיה זדונית, או מעלה קובץ שמכיל אותה, ומשנה לסוכן את המטרה — "בחלק הזה, הסוכן מתחיל לעבוד עבור התוקף. זה השלב הפתיחה של השרשיות התקפה שלנו" [15:16].

משם: Identity and Privilege Abuse, שבו התוקף מנצל את מנגנוני האמון של הסוכן כדי לשלוף נתונים שאינם מיועדים לו — "ממש כאילו בלבל אותו, confused deputy". אחר כך Memory & Context Poisoning, הזרקת נהלים ומדיניות שגויים לזיכרון ארוך-הטווח, כך ש"בשיחה הבאה של אותו תוקף זה עדיין יחזיר את אותו תשובות". ולבסוף פעולת API אמיתית שהסוכן מבצע עבור התוקף [15:16].

9. מהאיום ל-Threat Statement ול-STRIDE

כדי שהמתקפות יהפכו לפריטי עבודה, הן הוכנסו לתבנית קבועה בת ארבעה חלקים: מקור איום, עם הכשרה/תנאי מקדים מסוים, יכול לבצע פעולה, שמובילה להשפעה — ובסופה פגיעה בנכסים [16:48].



המשפט הממולא עבור היוזקייס הפיננסי, ומתחתיו סיווג STRIDE שבו הודגשו Spoofing ו-Information disclosure

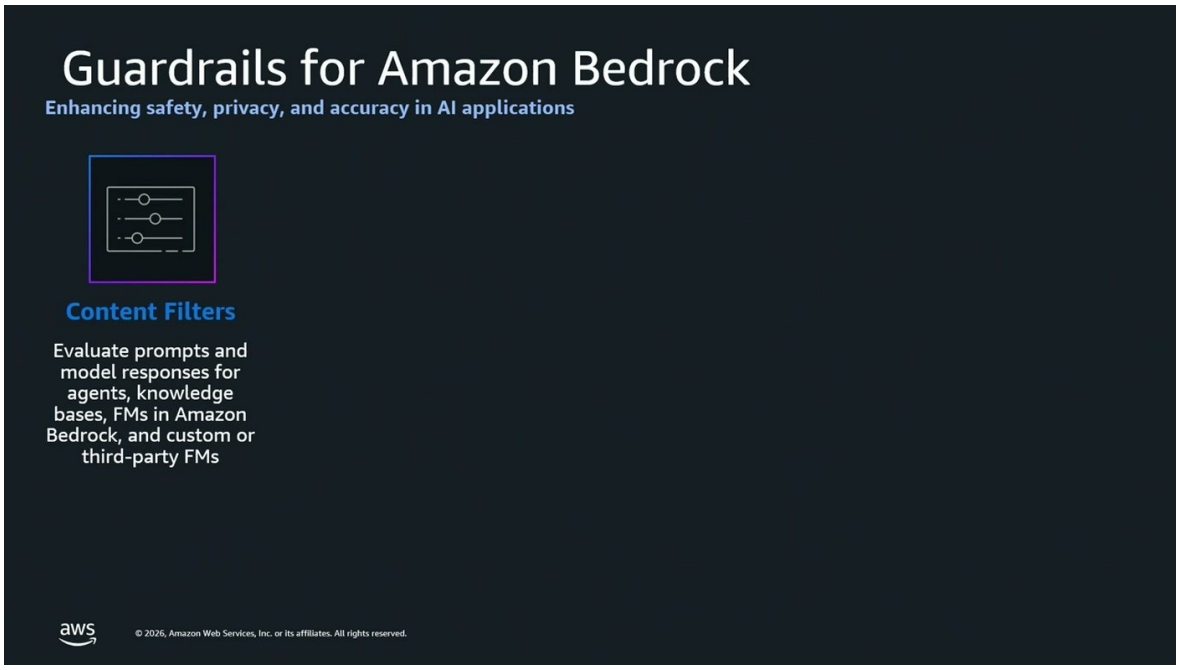
המשפט שנבנה על הבמה: שחקן זדוני בעל זהות IDP ולידית יכול להזריק פרומט זדוני ולשכתב את ה-system prompt של הסוכן, מה שמוביל לכך שהתוקף מקבל מידע פיננסי של לקוחות אחרים — ופוגע בחיסיון המידע בארגון [16:48]. הסיווג נעשה לפי STRIDE ("יש עוד כל מיני מודלים שאפשר להשתמש בהם, פה בחרנו בסטרייד"), והתעדוף נקבע HIGH "כי גם גנבו זהות גם אזריקו מידע שגוי לתוך הזיכרון וגם עשו טרנזקציות לא מורשות" [18:20].

סדרי גודל הערת מציאות מהבמה: "כשאתם תעשו תרד מודלינג על האפליקציה שלכם בעולם האמיתי יהיה לכם הרבה יותר משורה אחת יהיה פה איזה 30-40 אם לא 500 שורות" [16:48]. הטבלה בשקופית היא דוגמה, לא תוצר.

10. שאלה 3 — מה עושים לגבי זה

10.1 לפני המודל: Amazon Bedrock Guardrails

לפני שנכנסים בכלל ל-AgentCore, יש שלב מקדים. Guardrails תואר כ"מנגנון רב שכפתי שיושב בין המשתמש לבין המודל", שמאפשר לטפל בחלק מהפגיעויות "עוד לפני שזה מגיע בכלל למודל" [19:54]. הרכיבים שהוזכרו: content filters ו-prompt evaluation (שמטפלים ב-Agent Goal Hijack), denied topics, סינון PII ומניעת הזיות. נקודת ההשמה שהוצעה היא הלבשת ה-guardrail על AgentCore Gateway — כך הפנייה נחסמת לפני שהיא מגיעה לסוכן.



Content Filters מעריכים גם פרומטים וגם תשובות מודל — עבור סוכנים, knowledge bases, מודלים ב-Bedrock ומודלי צד שלישי

10.2 Amazon Bedrock AgentCore כסביבת ריצה מובטחת

AgentCore הוצג כ"היכולת שלכם בתור לוקחות AWS להריץ אפליקציות אייג'נטיות בסביבה מובטחת" [18:20], עם ארבעה רכיבים שנבחרו כי הם ממופים אחד-לאחד לארבע המתקפות. הדובר הדגיש שזו בחירה לצורך הסשן: "לאג'נט קור יש עוד המון יכולות אבל עכשיו התרכזנו בארבעת אלה" [19:54].

רכיב AgentCore	מה הוא נותן, כפי שהוצג	נגד איזו מתקפה
Identity	חיבור ל-identity provider חיצוני והחלטה באיזו זהות מופעל כלי	Identity and Privilege Abuse
Gateway	חיבור MCP וטיפול בכל הטוקנים שעוברים דרכו; גם נקודת ההשמה של ה-Guardrail	Agent Goal Hijack
Runtime	הרצת סוכן בכל פריימוורק, עם ההוראות המדויקות, רשימת הכלים המותרים וההקשר	Tool Misuse and Exploitation
Memory	זיכרון קצר וארוך טווח עם namespaces והפרדת אסטרטגיות	Memory & Context Poisoning

10.3 באיזו זהות מפעילים כלי — 2LO מול 3LO

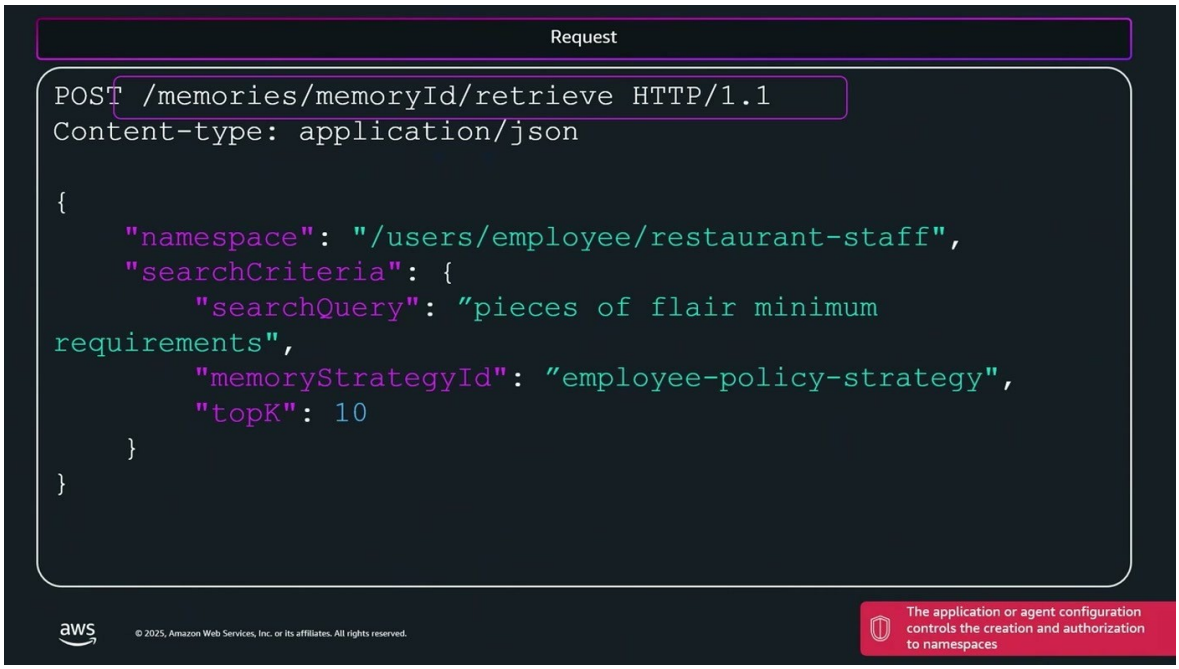
לפני בחירת המנגנון, הוצגו השאלות שקובעות אותו: מי בעלי המידע, האם יש משתמש שמבצע אינטראקציה עם הכלי, כמה זמן לוקחת ההפעלה (batch מול real time), ומה ה-[19:54] permission scope. משם שתי אפשרויות:

- **3LO —** "הסוכן פועל בשם המשתמש והמשתמש צריך בצורה אקטיבית לאשר לו לעשות את זה" [21:26]. בקוד: קריאת RequireAccessToken עם provider name (Azure Entra ID, Cognito, Okta ודומיהם), scope, ו-auth flow מסוג USER_FEDERATION — בפועל פופ-אפ שהמשתמש חייב לאשר.

- machine to machine — **2LO**: "הסוכן משתמש בזהות עצמו ופונה לאותו כלי". אותו מבנה קריאה, אותו scope, אבל auth flow מסוג M2M [21:26].

10.4 זיכרון: namespaces כגבול הדף

הזיכרון תואר כ"רכיב סופר קריטי" שמחולק לקצר טווח (אותו סשן) וארוך טווח (העדפות משתמש, סיכומי שיחות). אבל — והדובר הדגיש זאת — עצם הקיום של זיכרון מנוהל "זה עדיין לא מטפה לנו בממורי פויזנינג" [22:58]. הפתרון שהוצג הוא AgentCore Memory namespaces: היררכיה דמוית תיקיות, עם משתנים כמו actorId, strategyId ו-sessionId, שמאפשרת להגדיר את אסטרטגיית הזיכרון ברמת fine-grained.



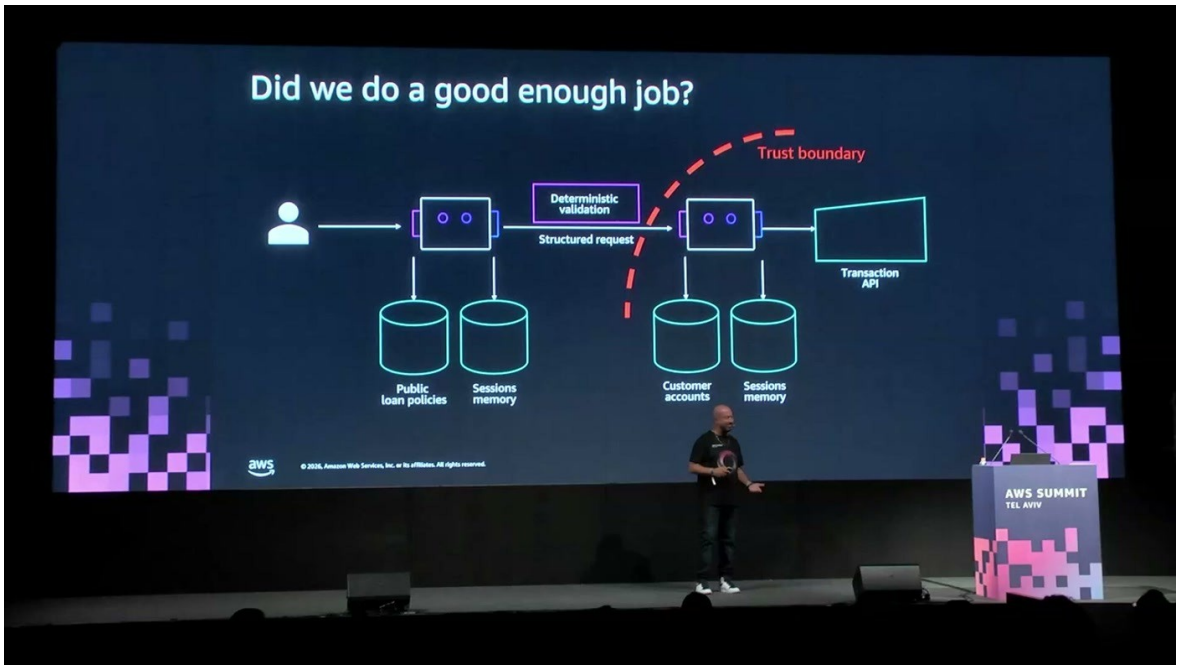
קריאת retrieve אמיתית: namespace היררכי, searchQuery בשפה חופשית, memoryStrategyId ו-topK של 10

ה-response מחזיר memory record id, את ה-namespace שממנו נשלף, את הטקסט, score רלוונטיות, זמן יצירה ו-strategy id [24:29]. הנקודה האבטחתית היא ההכלה: אם אחד ה-namespaces מורעל, "זה לא משפיע בהכרח על כלל הזיכרונות שיש לנו באותו ארגון" [24:29].

11. שאלה 4 — האם עשינו עבודה מספיק טובה

הסבב נסגר במיפוי חוזר: Agent Goal Hijack מטופל ב-Bedrock Guardrails ובהערכת הפרומט; Identity and Privilege Abuse מטופל בהחלטה מודעת בין 2LO ל-3LO; Memory & Context Poisoning מטופל ב-namespacing [24:29]. אבל התשובה לשאלה הרביעית הייתה "לא מספיק".

הסיבה נובעת מהיוזקייס עצמו: אותו סוכן נדרש גם למידע ציבורי (מדיניות הלוואות כללית) וגם לחשבונות לקוח. לכן הפתרון היה לפצל אותו לשניים — סוכן שעובד מול המידע הציבורי וסוכן שעובד מול חשבונות הלקוח, כל אחד עם אסטרטגיית זיכרון ו-namespaces משלו, כשביניהם עוברת בקשה מובנית שעברה deterministic validation.

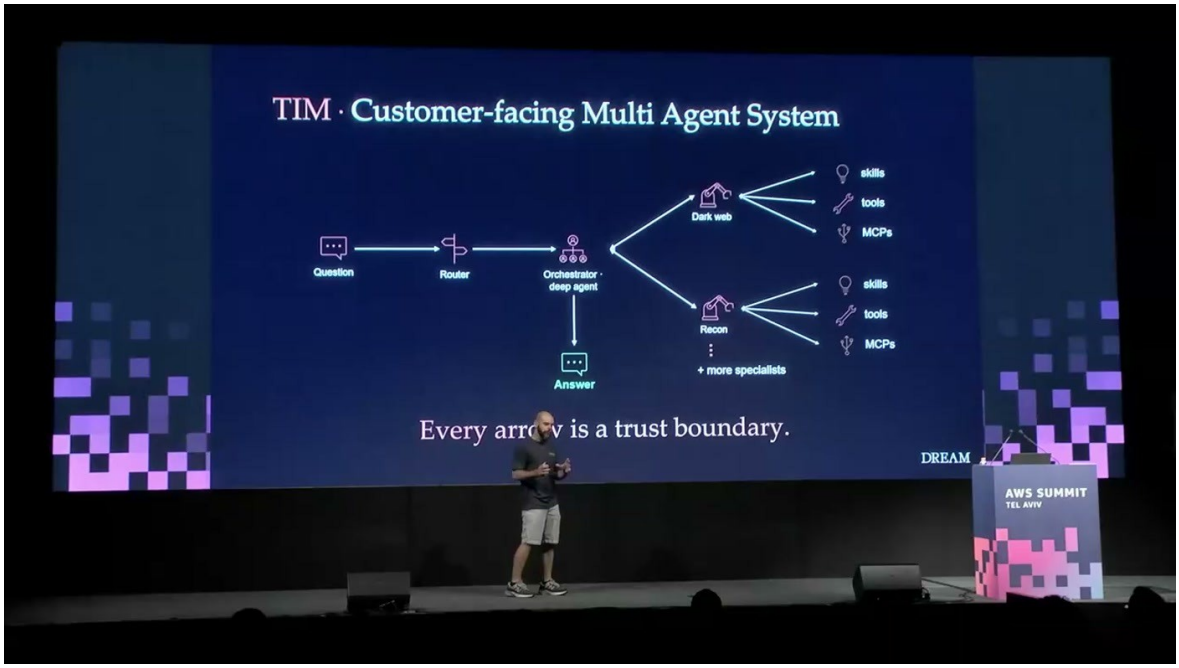


פיצול לשני סוכנים: ולידציה דטרמיניסטית ובקשה מובנית חוצות את גבול האמון, ורק הסוכן שמעברו ניגש ל-Transaction API

— ארכיטקט AWS, [26:01] המטרה היא באמת שאנחנו נהיה כמה שיותר דטרמיניסטיים. מה שעוזר לנו באמת להעלות את ה-Trust boundary בין הסוכנים שלנו

12. מהשטח: Dream, סוכנים בפרודקשן ובסקייל

החלק השלישי הועבר ל-AI/ML & Engineering Lead מחברת Dream, שעובדת מול ממשלות ומספקת Sovereign AI — "אנחנו נותנים להם את השליטה של ה-AI שלהם במדינות שלהם". הוא מוביל את קבוצת ה-CTI (Cyber Threat Intelligence), שמתמקדת ב"כל מה ש-dark web, clear web ומקורות חיצוניים, בלי out there": להסתכל על הרשת הפנימית של הלקוח [27:32].



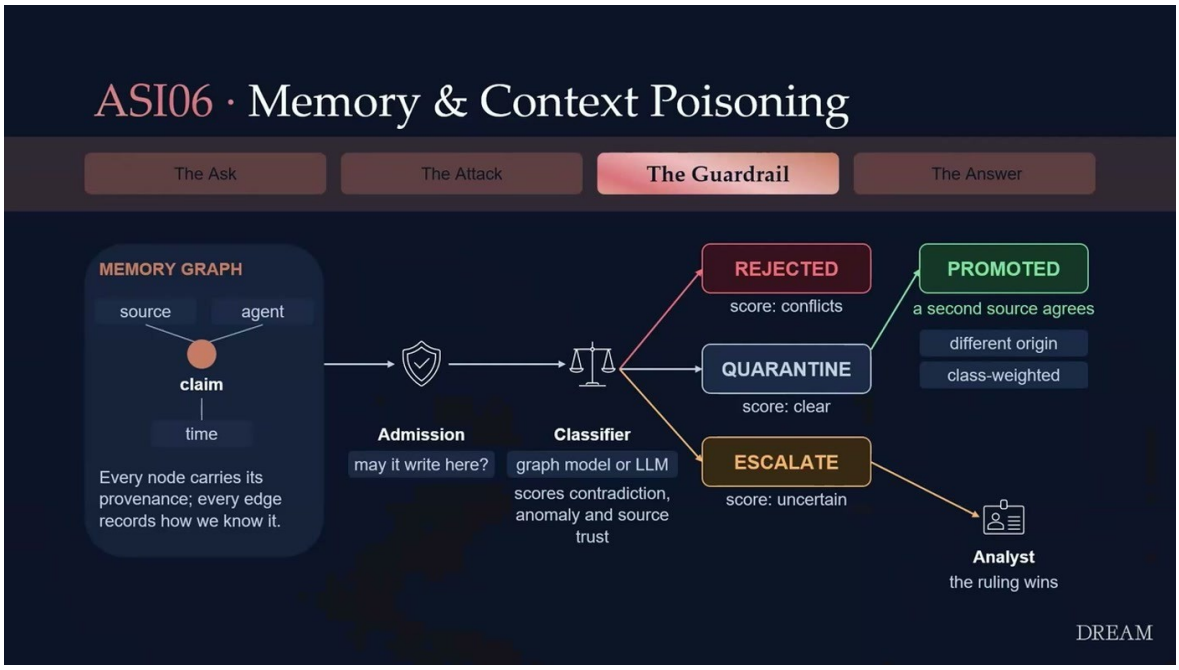
TIM: שאלה נכנסת, router מחליט אם בכלל יש כאן חקירת CTI, ו-deep agent מתזמר מומחים שלכל אחד skills, tools ו-MCPs משלו

הארכיטקטורה: פרומט מהלקוח מגיע ל-router, שמחליט אם יש בכלל מקום להיכנס ל-CTI או שאפשר לענות בלי כל ה"תירלול שקורה אחרי זה". אם כן — אורקסטרטור מסוג Deep Agent ("אני מניח שכל כך שמעתם על Deep Agent של Long Chain, אז זה לא זה, אבל דומה") עושה planning ו-to-do list ואז מאציל למומחים קטנים [29:03]. המסקנה שהובילה את שאר החלק:

> **AI/ML & Engineering Lead, Dream, [29:03] —** בכל חץ אנחנו יכולים לגרום לבעיות למצוא בעיות וכאלה בגלל זה אני קורא לכל חץ או איזשהו trust boundary שאנחנו צריכים להתייחס בזה

12.1 ASI06 — כשהזיכרון משקר

אנליסט של לקוח שואל: "have any of our credentials leaked on the dark web?". הסוכן סורק מקורות dark web ומוצא — "וואלה יש פה 12 אלף leaked credentials" [30:36]. אבל בהמשך הפוסט מופיעה הודעה קטנה "שכתוב זה לא באמת credentials אמיתיים זה synthetic data". השאלה היא למי להאמין, וההנחה היא שהתוקפים יודעים שסוכנים קוראים את זה: "אנחנו מניחים שהגמות תוקפים הם חכמים... הם יודעים איך גם לרמות אותם" [30:36].



כל טענה נכנסת כ-node עם provenance, עוברת admission ואז classifier — והתוצאה היא אחד מארבעה מסלולים

המנגנון: כל טענה נכנסת כ-node בגרף הזיכרון של הלקוח וכל קשר כ-edge, כולל המקור המדויק — מאיזה agent, מאיזה tenant, מאיזה משתמש. על הגרף רץ classifier, ובמכוון לא LLM-as-a-judge: "אני רוצה קצת יותר דטרמיניסטיות" [32:06]. לפי ה-score הטענה נדחית ולא נכנסת בכלל לקונטקסט ולא לזיכרון ("לא שורט טרם, לא לא טרם כלום"), או נכנסת ל-quarantine עד אימות ממקור נוסף, או מסולמת לאנליסט אנושי: "מרמים דגל, אין לי מושג מקווה פה, שמישהו יבוא ויציל אותי" [33:38].

התוצאה שהוחזרה ללקוח לא הייתה רק המספר, אלא גם ההקשר: יש 12 אלף credentials, והנה גם מי ניסה להטעות אותנו בדרך.

> **AI/ML & Engineering Lead, Dream, [33:38] —** בסופו של דבר אנחנו אומרים memory is evidence והוא לא authority אז אנחנו לא מאמינים לממורי blindly

12.2 ASI03 — כשהסוכן רוצה לעזור יותר מדי

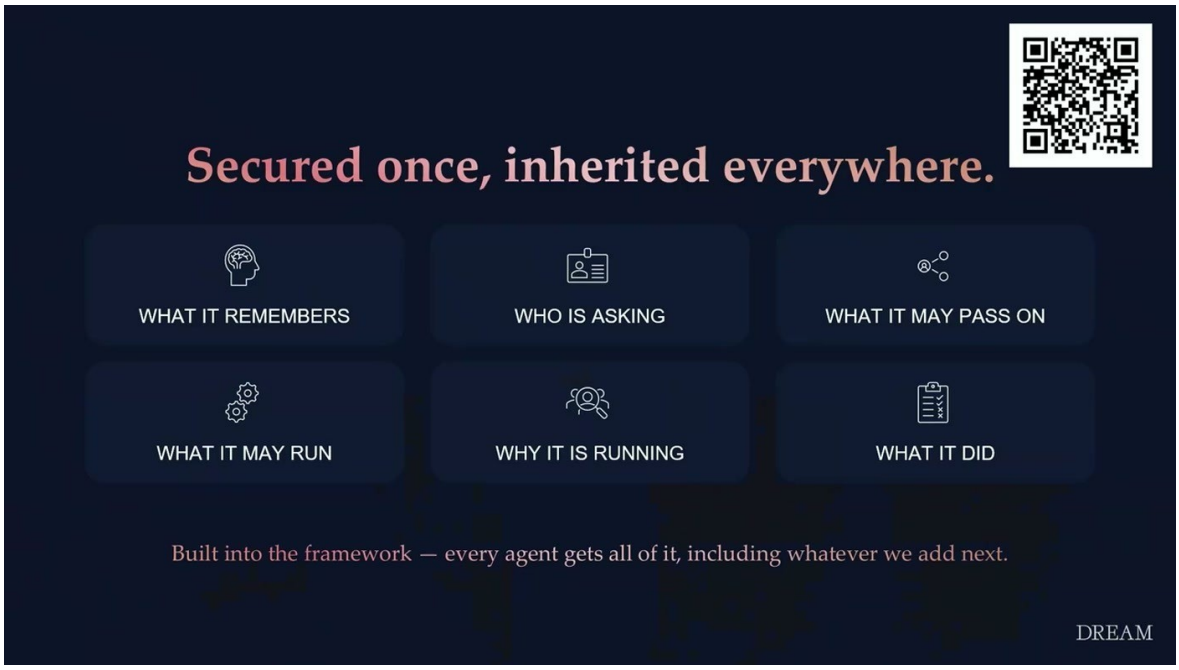
הדוגמה השנייה: אנליסט מבקש "compare our exposure against the rest of your customer base" [33:38]. הבקשה נשמעת לגיטימית, אבל משמעותה חשיפת מידע של לקוחות אחרים. הבעיה, בניסוח של הדובר: "אגנטים שלנו בעיקר בדרים הם מאוד חמודים... הם רוצים לעזור" — ולכן הם פשוט מתחילים לחפש [33:38].

מה שקורה מאחורי הקלעים: הבקשה מגיעה חתומה על ידי המשתמש, ידוע ה-tenant, הלקוח והתפקיד (אנליסט? CISO?). ברירת המחדל היא סירוב — "אנחנו קודם כל אומרים לא" — ומשם מחושב risk score על בסיס שאלות כמו האם המשתמש רשאי לשאול שאלה כזו, והאם הוא בדרך כלל שואל שאלות כאלה. שוב deny / allow / escalate [35:08]. ההדגשה החשובה היא שההחלטות לא מתקבלות רק ב-runtime: "הוא כבר עוד בעלייה של האגנט כבר החלטנו מה הוא יכול ומה הוא לא יכול" [35:08].

> **AI/ML & Engineering Lead, Dream, [36:41] —** ה-permissions של האג'נט הוא לא קשור ליוזר הוא קשור לאג'נט עצמו מה שאנחנו קנפקנו בהתחלה

12.3 איך זה עובד בסקייל: פריימוורק פנימי

הבעיה הארגונית שהוצגה: הרבה סוכנים, מפותחים על ידי קבוצות שונות, חלקם לוקחים אבטחה בחשבון וחלקם לא. התשובה הייתה פריימוורק פנימי בשם CTI Agency, שבו מוגדרים במקום אחד הזיכרון, האבטחה, ההרשאות והתפקידים של כל סוכן שמשתמש בו: "הכל מנועל במקום אחד. זה אומר שברגע שאנחנו מוצאים איזשהו אישו, באג'נט שלא קשור לזה בכלל, אנחנו יכולים לתקן את זה ברוט" [36:41].



שש ההחלטות שהפריימוורק כופה על כל סוכן — ומה שנוסף אליו בעתיד מגיע אוטומטית לכולם

הדובר ציין שהפריימוורק נמצא בתהליך פתיחה לקוד פתוח ("כל הדבר הזה עכשיו pending open source", [36:41]), תלוי באישורים משפטיים — ואף ביקש עזרה מהקהל בנושא.

13. מה לוקחים מכאן

בסיכום הוצג מודל פשוט להערכת סיכון של סוכן: "לא כל האגנטים נולדו שווים" [38:13]. שלושה גורמים קובעים את רמת הסיכון, וצירוף שלהם מעלה אותה:

- **חשיפה לדאטה לא מהימן —** "עד כמה הוא חשוף לאן טראסטי דייטא?" [38:13]. ככל שיותר מקורות חיצוניים נכנסים לקונטקסט, כך גדל הסיכון.
- **רמת האג'נסי —** "עד כמה יש לו יכולת לזום תהליכים או לבצע פעולות ללא בקרה אנושית" [38:13].
- **ערך הנכסים שאליהם הוא חשוף —** "ככל שהנכסים הם יותר high value, ככה רמת הריסק עולה" [38:13].

רשימת פעולות לצוות שבונה סוכן

- **להגדיר scope מדויק לפני שכותבים קוד —** לאיזה דאטה, אילו כלים ואילו יכולות הסוכן באמת נדרש [39:47].
- **להכריע במפורש בשאלת הזהות —** זהות הסוכן מול זהות המשתמש, ולזכור שלאותו משתמש ביוזקייסים שונים יכולות להיות הרשאות שונות [39:47].
- **למקם את בקרות ההרשאות והזהות מחוץ למודל —** "ולא להסתמך על האוטפוט שלו כי אנחנו רוצים קונטרולים דטרמיניסטיים" [39:47].
- **לפצל סוכן שמשרת יותר מרמת רגישות אחת —** כפי שהודגם בפיצול סוכן המידע הציבורי מסוכן חשבונות הלקוח, עם ולידציה דטרמיניסטית על הגבול.
- **להפריד namespaces בזיכרון —** כדי שהרעלה של אזור זיכרון אחד לא תתפשט לכלל הארגון.
- **לא להאמין לזיכרון בעיוורון —** לשמור provenance לכל טענה, ולהריץ עליה בדיקה דטרמיניסטית לפני שהיא הופכת לידע.
- **להריץ threat modeling על כל הפלואו מקצה לקצה —** "אפליקציה האג'נטית היא קודם כל אפליקציה ולכן חשוב להגן על כל הפלואו מקצה לקצה" [39:47].

פערים מה הסשן לא כיסה, ושווה לשים לב אליו: מיפוי data flow מלא (הוזכר ונדחה מחוסר זמן), שישה מתוך עשרת סיכוני ה-OWASP, ועלויות או השלכות ביצועים של המיטיגציות שהוצגו. גם היוזקייס הפיננסי הוא דוגמה מלאכותית ולא מערכת שרצה בפרודקשן — רק החלק של Dream הוא מהשטח.

מילון מונחים

מונח	פירוש
Agent Goal Hijack (ASI01)	שינוי מטרת הסוכן באמצעות הנחיה זדונית שהוזרקה לפרומט או לקובץ שהועלה
Identity and Privilege Abuse (ASI03)	ניצול הרשאות ויחסי אמון של הסוכן לשליפת מידע שלא נועד למבקש; גרסה אג'נטית של confused deputy
Memory & Context Poisoning (ASI06)	הזרקת מידע שגוי לזיכרון ארוך-טווח, כך שהשפעתו נמשכת גם לשיחות הבאות
Bedrock Guardrails	שכבת סינון בין המשתמש למודל: content filters, prompt evaluation, denied topics וסינון PII
AgentCore Gateway	נקודת חיבור מנוהלת לכלים ול-MCP, שמטפלת בטוקנים ומשמשת גם כנקודת אכיפה ל-guardrail
AgentCore Identity	רכיב שמחבר identity provider וקובע באיזו זהות מופעל כלי
2LO / 3LO	שתי דרכי הזדהות להפעלת כלי: 2LO הוא machine-to-machine בזהות הסוכן, 3LO הוא פעולה בשם המשתמש הדורשת אישור אקטיבי שלו
Memory namespace	נתיב היררכי שמארגן זיכרונות מחולצים, כולל משתנים כמו actorId ו-sessionId, ומשמש גם לבקרת גישה והכלה
STRIDE	מודל סיווג איומים: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege
Threat statement	ניסוח קבוע של איום: מקור איום, תנאי מקדים, פעולה, השפעה, ונכסים שנפגעים
MCP	פרוטוקול פתוח לחיבור מערכות AI לכלים, מקורות מידע ושירותים חיצוניים
Deep Agent	סוכן מתזמר שמבצע planning ו-to-do list ורץ לאורך זמן, ומאציל משימות לסוכנים מומחים
CTI	Cyber Threat Intelligence — איסוף וניתוח מודיעין איומים ממקורות חיצוניים לארגון