סמינר במדעי המחשב - סיכום
מגיש: קונסטנטין קזקוב
שם המאמר: Deceiving Cyber Adversaries: A Game Theoretic Approach
המחברים: Aaron Schlenker, Fei Fang, Milind Tambe, Long Tran-Thanh, Phebe Vayanos, Yevgeniy Vorobeychik
כנס: AAMAS 2018
קישור: http://ifaamas.org/Proceedings/aamas2018/pdfs/p892.pdf

הבעיה שהמאמר בא לפתור: להקשות על תוקפי סייבר ע"י שילוב של תגובות אמיתיות, שקריות וסתומות במענה בסריקת הרשת שנעשת ע"י כלים כמו NMap שעונים על השאלות "איזה מחשבים מחוברים אחד לשני, כתובות שלהם, אילו מערכות הפעלה הם מריצים" ועוד, וכיצד צריך המגן להסוות את המערכות כדי לגרום ליריב לתקוף את המערכות הפחות יקרות?

המצב כיום: מנהלי רשתות משתמשים בטכניקות כגון "רשימה לבנה" של יישומים(whitelisting), נעילת הרשאות ותיקון נקודות תורפה באופן מיידי. מחקרים קודמים מראים ששימוש במערכות הטעיה נותן יתרון משמעותי למגן בכך שבזמן שהתוקף "מבזבז" זמן ומשאבים על "חלת הדבש"(שרת/מחשב שעושים עבודה שלא באמת קשורה לרשת של החברה), המגן יכול לבחור האם לחזק את רמת ההטעיה או לעצור אותה(כדאי שהתוקף לא יחשוד בכך שהוא השיג מידע לא אמין). עם זאת, חסרון אחד של רוב הגישות הקודמות הוא שהם לא מספקים מודל כראוי אופי היריב של תחום אבטחת סייבר.

הפתרון שהמאמר מציע:המאמר מציג מודל חדש (תיאורטי) של אינטראקציות מטעות מסוג הטעיה בתגובות לסריקות רשת בין מגן לבין תוקף סייבר, שמכנים אותו 'משחק הונאה של סייבר'.
משחק ההונאה של סייבר הוא דואופול סטקלברג בין מנהל הרשת(המגן) לבין ההאקר(התוקף) כאשר המגן משחק ראשון ובוחר כיצד המערכת/רשת תגיב לשאילתות מההאקר, ולאחר מכן ההאקר בוחר איזו מערכת/רשת לתקוף(בהתאם לתור של המגן), מרכיביו השונים של המשחק ומאפייני המודל הנ"ל מתוארים בפירוט כדלקמן:
מערכות ותצורות אמיתיות(מה יש בפועל), תצורות שנצפו(מה אנחנו רוצים שהוא יראה), אסטרטגיית המגן, אסטרטגיית התוקף, תועלת(כמה התוקף מרוויח אם יתקוף תצורה ספציפית).
לאחר שהוגדרו משתני המשחק, נגדיר שני שחקנים, שחקן חזק שיודע את הנתונים תצורות אמיתיות, נצפות, מטריצת האילוצים, תועלת ואת האסטרטגיה שלנו ובוחר לתקוף את התצורה הנצפת עם התועלת הכי גבוהה, אם המגן בוחר באסטרטגיה הממזערת את התועלת הצפויה של שחקן חזק, הוא מקבל אסטרטגיה חזקה כמגן ויכול להיות בטוח כי לא משנה כמות הידע של היריב, אין אסטרטגית משחק שיכולה להוביל לאובדן גדול יותר עבור המגן, בהתאם לעקרון המינימקס, ושחקן נאיבי שפועל רק לפי מה שהוא רואה. בעזרת רידוקציה כותבי המאמר הוכיחו שהבעיה היא NP קשה(בעיית החלוקה) למציאת אסטרטגיה אופטימלית נגד שחקן חזק, למרות זאת הם מביאים אלגוריתם חמדני של מינימקס שעובד מהר ומביא תוצאה טובה בממוצע.
<u>האלגוריתם החמדני מינימקס</u> מקבל את התצורות האמיתיות, הנצפות, התועלות והעלות(במידה ויש) ומוציא מטריצת האסטרטגייה של המגן בגודל כמות התצורות האמיתיות על כמות התצורות הנצפות.
דוגמא לריצה כזו: יש לנו 4 תצורות אמיתיות עם התועלות 10,0,0,5 , ושתי תצורות נצפות, באופן אקראי ניקח את שלושת התצורות האמיתיות ונצרפן לתצורה הנצפת הראשונה, נחשב את התועלת לתצורה זו ונקבל 3.33(ממוצע) עבור התצורה הראשונה, נצרף את התצורה האחרונה לתצורה הנצפת השנייה, ונקבל תועלת 5. נשמור תוצאה זו ונמשיך הלאה באופן אקראי(תוצאה טובה יותר מתקבלת ברגע שיש יותר איטרציות). באיטרציה השנייה נצרף את התצורה הראשונה לנצפת הראשונה, וכל השאר לשנייה, ונקבל תועלות 10 ו1.67, תוצאה יותר גרועה לכן נשמור את הראשונה כמועדפת.
האלגוריתם החמדני של המגן בהנחה שהשחקן הוא חזק, עלול לגרום למגן לא להבין את היתרון האינפורמטיבי כאשר מתמודד מול שחקן נאיבי, במקרה הזה ערכי התועלת הם קבועים והאסטרטגיה של המגן אינה משפיעה על התועלת הצפויה של היריב לתקוף תצורות נצפות שונות. אלגוריתם זה יעיל ומהיר בהינתן שאין הגבלת תקציב, ברגע שמכניסית הגבלת תקציב הבעיה הופכת להיות NP קשה(רדוקציה לבעיית התרמיל).
דוגמא לריצה של אלגוריתם כזה: יש לנו 3 תצורות אמיתיות עם התועלות 5,10,0 ושני תצורות נצפות עם תועלות נצפות 10 ו5, באיטרציה הראשונה באופן אקראי נצרף את כל התצורות לתצורה הראשונה ונקבל

תועלת מינימלית 5, באיטרציה הבאה נצרף את שני הראשונות לשנייה ואת השלישית לראשונה ונקבל תועלת מינימלית 0, וכך קיבלנו את האסטרטגייה הטובה למקרה זה.

Algorithm 1: Greedy-Minimax

```
minIndCost[] ← (min_f̃ c(f, f̃))_{f∈F}
minTotCost ← Σ_f N_f * minIndCost[f]
initialize minu*, σ_best
For iter = 1 ... numIter
    K_list[] ← shuffle(K)
    initialize remB ← B, reqB ← minTotCost
    initialize σ[], Ñ[], Ū[]
    For i = 1 ... |K|
        k ← K_list[i], f ← x[k]
        σ[k] ← GMMAssign(f, σ[], Ñ, Ū[], remB, reqB)
        Ñ[σ[k]] ← Ñ[σ[k]] + 1
        update(Ū[σ[k]])
        remB ← remB − c(f, σ[k])
        reqB ← reqB − minIndCost[f]
    compute u* = max_f̃ Ū[f̃]
    update(minu*, u*, σ_best, σ)
return σ_best
Procedure GMMAssign(f, σ[], Ñ, Ū[])
    initialize newU*[]
    For f̃ ∈ F̃_f
        If (reqB − minIndCost[f] + c(f, f̃) > remB) Then
            Continue
        σ[k] ← f̃
        newU*[f̃] ← U*(σ)
    F̃_best ← argmin_f̃ newU*[f̃]
    generate f̃_best ~ uniRand(F̃_best)
return f̃_best
```

Algorithm 2: Compute defender's optimal ϕ with fixed $\bar{U}_{\tilde{f}}$.

```
initialize φ, Γ*, f̃*
sort(F̃) //descending by utility Ū_f̃
minUtil[] := (min_f̃ Ū_f̃)_f
For i = 1, ..., |F̃|
    initialize Γ'
    P1:={f | minUtil[f] > Ū_f̃i}
    If P1 ≠ ∅
        break
    P2:={f | minUtil[f] = Ū_f̃i}
    P3:={f | minUtil[f] < Ū_f̃i and f̃i ∈ F̃_f}
    P4:={f | minUtil[f] < Ū_f̃i and f̃i ∉ F̃_f}
    Γ' := P2
    update(Γ', P3)
    update(Γ*,Γ',f̃*,f̃i)
update(φ,Γ*,f̃*)
return φ
```

בדיקת הפתרון: המחברים מעריכים את מודל המשחק והפתרון באופן מלאכותי, תמורות המשחק כסכום אפס(בהתאם למודל של סטקלברג), ולכל תצורה אמיתית התמורה מתפזרת באופן אחיד בין 1 ל10. לכל תצורה נצפת מוקצה באופן אקראיקבוצה של תצורות אמיתיות שיכולות להיות נסתרות על ידה, כאשר מובטח לנו שכל תצורת אמיתית יכולה להיות מוסתרת לפחות ע"י תצורת נצפת אחת. העלויות של יצירה מתפזרת באופן אחיד בין 1 ל100 אם תקציב בגודל B שמתפזר באופן אחיד בין העלות המינימלית והעלות המקסימלית שניתן להקצות. בכל הניסויים יש מופעים אקראיים בממוצע ביותר מ30 וריאציות. התוצאות הראו שהאלגוריתם החמדני רץ תמיד פחות משנייה אחת לכל כמות התצורות הנצפות, בזמן שהאלגוריתם המלא(NP קשה) לכל קפיצה של כמות התצורות, זמן הריצה שלו גדל יחד, התוצאות היו לטובת האלגוריתם המלא,אך הפרשי התוצאות הם נורא קטנים(כלומר העלות "שנוזקה" מהאלגוריתם החמדני גדולה בקצת מהעלות "שנוזקה" מהאלגוריתם המלא).
תוצאות נוספות הן כיצד האלגוריתמים רצים עבור שני סוגי השחקנים(שחקן חזק ושחקן נאיבי), והתוצאות הן שכאשר מריצים את האלגוריתם החמדני מול השחקן הנאיבי אנו מאבדים הרבה יותר מאשר אם היינו מריצים את האלגוריתם השני(עבור הנאיבי), וכאשר היינו מריצים את האלגוריתם החמדני מול השחקן החזק, היינו מפסידים פחות מאשר אם היינו מפעילים את האלגוריתם השני(הנאיבי).

נושאים לשיפור בעתיד: המחברים מציעים לפתור את שני הבעיות שהם נתקלו בהם, אחת מהן היא ההנחה שלהן שההאקר תוקף רק מערכת אחת, כאשר בפועל בעולם האמיתי האקרים יכולים לתקוף מערכות מרובות, והשניה היא ההנחה שהמגן רואה רק שחקן חזק או שחקן עם קבוצה קבוצה של העדפות, כאשר במציאות הידע של ההאקר טמון בין שני הקצוות הללו. לעבודה עתידית יהיה חשוב להראות מודל שבו התוקף מכיל מידע חלקי, וכיצד זה משפיע על תגובת המגן.

דעתי על המאמר:די מובן מהמאמר למה המחברים לא יכולים לבדוק את המודל בעולם האמיתי(הם ציינו גם למה), לכן אין כאן תלונות לכך שאין סטטיסטיקות אמיתיות, אבל היה כדאי להוסיף אולי למודל פרמטרים נוספים, כמו זמן התגובה בין כל תור של השחקנים במודל, ואולי אפילו הגבלת זמן למציאת האסטרטגיה האופטימלית, כי בעולם האמיתי הכל עניין של זמן תגובה למצב.
בנוסף, המחברים התבססו הרבה על מאמרים קודמים, אך ללא שימוש בהם במודל שלהם(לדוגמא מחשב/שרת שמדמה שרת חשוב אך בפועל הוא פשוט מוסר מידע מזויף, לעומת זיוף שאילתות).