# Review 28: Kitsune: An Ensemble of AutoEncoders for Online Network Intrusion Detection

**Paper: https://arxiv.org/abs/1802.09089v2**

פינת הסוקר:

המלצת קריאה מעדן ומייק: מומלץ לאנשים העוסקים בתחום ה-NIDS או לאנשים שאוהבים Anomaly Detection ו-Autoencoders

בהירות כתיבה: בינונית

ידע מוקדם: Autoencoders, Online Algorithms

יישומים פרקטיים אפשריים: Network Intrusion Detection System

פרטי מאמר:

פורסם בתאריך: 25.02.18, בארקיב.

הוצג בכנס: Network and Distributed Systems Security Symposium (NDSS) 2018.

תחומי מאמר:

זיהוי אנומליות

שימוש ואימון של אלגוריתמים בזמן אמת (Online)

למידת אנסמבל

כלים מתמטיים, מושגים וסימונים:

Autoencoders

מבוא:

התקפות הסייבר רק הולכות ונעשות מתוחכמות יותר בשנים האחרונות. אחד מכלי ההגנה הנפוצים הינו NIDS - Network Intrusion Detection System. כלי זה הוא שם למערכת תוכנה או חומרה אשר סורקת את הרשת בחיפוש אחר תעבורת רשת זדונית שיכולה להעיד על התקפה. כאשר תעבורה מסוג זה מזוהה, המערכת מעלה התרעה לאחראי המערכת כדי לידע אותו על המתרחש. כדי שתוקפים לא יעקפו את המערכת היא לרוב מחולקת באופן מבוזר ברשת הארגון במיקומים שונים (למשל בתוך הרשת, בחיבור של הרשת עם האינטרנט וכדומה).

כבר שנים רבות נעשה שימוש באלגוריתמי למידת מכונה עבור NIDS. רשת ניורונים הפגינו ביצועים חזקים בתור NIDS בשל היכולת ללמוד דפוסים מורכבים. הגישה הנאיבית אומרת שאם ניתן לרשת לראות מידע רגיל של הרשת הארגונית ומידע שנוצר מהתקפת סייבר, הרשת תלמוד להפריד בין שני סוגי תעבורה הנ"ל. בנוסף כדי לאפשר את הגישה המבוזרת בצורה קלה וזולה, הדבר ההגיוני לעשות הוא לשלב את המודל לתוך החומרה עצמה בנתבים היושבים ברשת. אולם לגישה זו יש כמה חולשות:

צורך בשמירת מידע רב על הנתבים: באימון מודל לא מקוון(רגיל) יש צורך לאסוף כמות מידע מסוימת לאורך הזמן ולאחר שנאספה כמות מספקת, המודל מאומן על הדאטה שנאסף. שיטה זו צורכת משאבים רבים יותר (כגון זיכרון וזמן CPU), מאשר המקבילה שלה - אימון מקוון. צריכת המשאבים הגבוהה עלולה לפגוע בביצועי הנתבים וכתוצאה מכך לפגוע ברשת הארגונית אם מאמנים את המודל בצורה לא מכוונת

דפוסי התקפות משתנים מקשה על ביצוע למידה מפוקחת ״סטטית״ - אימון מודל עם הגישות המפוקחות מצריך מידע מתויג. השגת מידע מתויג איכותי במקרה זה היא משימה לא טריוויאלית, שכן כיוון שההתקפות כל הזמן משתנות; המודל שמאומן על מידע היסטורי עלול להתקשות לזהות דפוסי תקיפות חדשים. בנוסף ההתקפה יכולה להיות כבר לא רלוונטית זמן קצר לאחר השגת המידע המתויג שכן התוקפים משכללים את ההתקפות שלהם כל הזמן.

סיבוכיות חישוב - סיבוכיות החישוב גבוהה של רשת נוירונים עלולה להכביד על נתב לבצע את ההסקה (inference) בזמן אמת.

הערה: באימון מקוון המודל מתעדכן בצורה שוטפת כלומר כל דגימה (או סט של דגימות) משמשת לעדכון של פרמטרי הרשת.

בעבר הוצעו כמה שיטות לזיהוי אנומליות בתעבורת הרשת. שיטה מקוונת בשם PAYL IDS המבוססת על חישוב היסטוגרמות של מאפיינים שונים של חבילת (packet) המידע. גישה זו הינה חסכונית מבחינת המשאבים הנדרשים ויכולה בקלות לרוץ על נתבים חלשים, אך יחד עם זאת סובלת מביצועים חלשים. גישה נוספת משתמשת במספר רשתות נוירונים שונות, שכל אחת מהן מאומנת לזהות התקפה ספציפית. גישה אלו מניחה למידה מפוקחת(supervised) ובנוסף דורשת שמירה של סט האימון על המכשיר המקומי, הנחה שהיא לא תמיד מתקיימת כשזה מגיע לציוד רשת כמו נתבים.

כדי להתגבר על קשיים אלו, המאמר מציג את שיטת Kitsune, שהיא שיטת זיהוי אנומליות המשלבת אנסמבל של (Autoencoders (AE. השיטה מסוגלת ללמוד בזמן אמת, בצורה לא מפוקחת וצורכת מעט משאבים תוך כדי הצגת תוצאות גבוהות לא פחות משיטות לא מקוונות קודמות (Offline).

הרעיון הכללי מאחורי המאמר:

לפי השיטה כל מאפיין (feature) ממופה לאחד ה-AE בשכבת האנסמבל. כל AE מבצע את משימת שחזור הקלט ומחשב את שגיאת השחזור, כלומר את ה-RMSE. אותו RMSE הוא הפלט של כל אנסמבל. ה-RMSE של כל רשת מועבר כקלט למודל נוסף של AE שמקבל את ה-RMSE-ים של כלל האנסמבלים כקלט ומנסה ללמוד את ההתנהגותם הרגילה בזמן האימון. בכך ה-AE לומד את הקשר בין האנסמבלים השונים. הפלט של ה-AE האחרון הינו ציון בודד שמציין עד כמה הדגימה נחשבת כאנומלית.

הכותבים מדגישים שבמהלך האימון רק דגימה אחת נשמרת בזיכרון לכל אורך תהליך האימון. זאת על ידי שימוש ב-SGD,כאשר backpropagation מופעל על דגימה בודדת במקום על אוסף של דגימות (batch). לאחר עדכון של משקלי הרשת הדגימה נזרקת ובכך שומרים על האלגוריתם כמקוון. כדי להתמודד עם סיבוכיות החישוב הגדולה של AE, כותבי המאמר הגבילו את גודל הרשת המקסימלית ל-3 שכבות עם מקסימום 7 נוירונים בשכבת הקלט והפלט. בכך ניסו הכותבים לאפשר לרשת לרוץ ללא מאמץ על נתבים.

כעת נרחיב על המבנה של השיטה המוצעת:

מחלץ המאפיינים (FE):

כותבי המאמר הדגישו כי הגישה הנאיבית הינה שימוש בחלון עליו מחושבים המאפיינים הסטטיסטים כגון מקסימום, ממוצע וכדומה. למשל בחלון של 5 שניות מחשבים את ממוצע ה-PAYLOAD של כל חבילה. הבעיה עם גישה נאיבית זו היא שקצב הגעת המידע יכול להיות כה גבוה שהחלון יכיל מספר גדול מאוד של דגימות ויעמיס על הנתב מאחר וכמות גדולה של דאטה צריכה להיות מאוחסנת עליו (הנתב). לכן הם הציעו להשתמש בסטטיסטיקה מצטברת דועכת (Damped Incremental Statistics) שבה בכל עת נשמר בזיכרון סט של המאפיינים הבאים: כאשר שלושת המדדים הינם: כמות הדגימות, הסכום של ערכי הדגימות וסכום הריבועים של ערכיהן. בכך לדגימה חדשה יהיה ניתן לבצע את העדכון הבא: ולאחר מכן לחשב את כלל המאפיינים החדשים:

בתהליך זה יש גם חשיבות לזמן שכן התנהגות הרשת הארגונית משתנה לאורך זמן כך המודלים צריכים לשנות את ההתנהגות שלהם בהתאם לשינויים של הקלט. כדי לבצע זאת משתמשים כותבי המאמר בפונקציית דעיכה:

כאשר הינו קבוע הדעיכה ו-t הוא הזמן שעבר מאז הדגימה האחרונה. כלומר השפעה של דגימה על הפרמטרים של המודל תדעך עם הזמן. ט המאפיינים החדש מוגדר באופן הבא: כאשר הינו המשקול הנוכחי, הזמן של העדכון האחרון ו- הינו סכום השגיאות (Residual sum) בין שני ערוצי התקשורת i,j. במהליך האימון וההרצה של המערכת מתעדכן סט המאפיינים הנ"ל באמצעות מקדם הדעיכה:

על ידי שימוש בממוצע וסטיית התקן חושבו מאפיינים נוספים בין ערוצי תקשורת שונים כגון השונות בין הערוצים, גודל הערוצים וכדומה. כלל מאפיינים אלו (יחד עם הממוצע וסטיית התקן של כל ערוץ) הם המאפיינים שנכנסו למודלים השונים בתור הקלט:

בסוף פיצ'רים אלו נבנים מעל 5 מקטעים של זמן ובסך הכל מימד הקלט לממפה מאפיינים הוא 115.

ממפה מאפיינים (Feature Mapper):

תפקיד רכיב זה הינו למפות תת-קבוצת מאפיינים מהקלט לאחד ממודליי AD. כדי לדעת כיצד לחלק את מרחב המאפיינים של הקלט בצורה טובה, נעשה שימוש באלגוריתם אשכולות היררכי (clustering). הקליסוטור מבוסס על הקורלציה בין מאפיינים (המרחק בין מאפיינים הינו ביחס הפוך לקורלציה ביניהם). המטרה היא לבנות k אשכולות כך שבכל אשכול לא יהיו יותר מ-m מאפיינים כאשר m הוא מספר הנוירונים בשכבת הקלט של כל AD. כיוון שמרחב המאפיינים הינו קטן ניתן לבצע את החישוב הזה בזמן ריצה.

מזהה אנומליות (Anomaly Detector):

רכיב זה נקרא KitNET והוא מורכב משתי שכבות כפי שניתן לראות בציור 3: שכבת האנסמבל ושכבת הפלט, כל אחת מורכבת מאוטו-אנקודרים.

שכבת האנסמבל: השכבה הראשונה ב-KitNET נקראת שכבת האנסמבל. שכבת האנסמבל מורכבת ממספר אוטו-אנקודרים כאשר כל אחד מהם מחשב את ה-RMSE עבור הקלט אותו קיבל ממפה המאפיינים. הפלט של כל אנסמבל עובר כקלט לשכבה הבאה.

שכבת הפלט: שכבה זו כוללת אוטו-אנקודר יחיד שמקבל את כלל ה-RMSE מכל האנסמבלים. שכבה זו פולטת לבסוף RMSE בודד שהוא ציון האנומליה של אותה דגימה.

הערה: הספריות החיצונית בציור הינן ספריות שכותבי המאמר השתמשו בהן ללא שינוי ולכן לא הוזכרו.

הסבר של רעיונות בסיסיים:

Autoencoder הינו מודל שמטרתו היא לדחוס את הקלט כך שיהיה ניתן לשחזרו מצורתו הדחוסה (הנקראת לפעמים ייצוג לטנטי או הקוד של הקלט) בדיוק מקסימלי. במילים אחרות אוטו-אנקודר מנסה לחלץ מהקלט את המאפיינים המהותיים שלו תוך כדי ניפוי מאפיינים פחות חשובים (redundant). אוטו-אנקודר מורכב שתי רשתות המופעלות אחת אחרי השנייה. הרשת הראשון, הנקראת אנקודר או מקודד, דוחסת את הקלט (מקטינה את מימדו) ומפיקה ממנו את הייצוג הלטנטי. החלק השני נקרא decoder או מפענח שמטרתו לשחזר את הקלט מהייצוג הלטנטי המופק על ידי האנקודר. אוטו-אנקודר מאומן למזער את שגיעת שחזור הקלט.

במקרה שלנו אוטו-אנקודר מנסה לאמן רשת לשחזר את מאפייני הקלט. הכותבים השתמשו בפונקציית Root Mean Square Error (או RMSE) בתור פונקציית לוס (מרחק בין הוקטור המשוחזר לקלט).

הישגי מאמר:

לצורך בחינת ביצועי Kitsune כותבי המאמר ניסו לבדוק האם המערכת תזהה התקפות המבוצעות על מצלמות IP ולשם כך הקימו רשת המכילה כמה מצלמות מסוג זה המחוברות לאתר מרוחק דרך VPN. דרך אותו VPN המשתמש יוכל לגשת למצלמות. מערכת Kitsune נפרסה בנקודת הגישה למצלמות. דרך נקודה עלול התוקף לגשת למצלמות. בנוסף כדי לבחון את Kitsune על רשת רועשת יותר המדמה יותר את העולם האמיתי, הוקמה רשת המכילה מכשירי IoT שונים יחד עם שלושה מחשבים רגילים. ברשת זאת אחת המצלמות הודבקה מראש בנוזקה כדי לבדוק את ביצועי המערכת מולה.

השוואה:

ביצועי המערכת הושוו מול ביצועים של מודלים מסוג לא מקוון כגון: Isolation Forest, Gaussian Mixture Models (כלומר כאלו שיש להם גישה לכל סט האימון) ומול מודלים מקוונים (עם גישה לדגימה אחת בכל פעם) : Incremental GMM, pcStream2. בנוסף נעשה שימוש במערכת NIDS Open source בשם Suricata שהיא מערכת שנעשה בה שימוש בתעשיה, זאת כדי לבדוק את ביצועי המערכת במאמר מול מערכת אמיתית.

כותבי המאמר השתמשו בגדלים שונים של שכבת הקלט של כל AE על ידי שינוי פרמטר m של המערכת מ-1 עד 10. כמה מהמדדים החשובים בהם השתמשו הינם True Positive Rate (TPR) PR, False Positive Rate(FPR), AUC:

ניתן לראות שלמרות שלמודלים הלא מקוונים הייתה גישה לכל סט האימון, ביצועי Kitsune לא נפלה מהם ואף בחלק מההתקפות גברה עליהם. בנוסף ניתן לראות בבירור ש-Kitsune מנצחת את כלל המתחרים המקוונים כמעט בכל התקפה, כפי שניתן לראות מתוצאות ה-AUC בהם Kitsune (ימין קיצון) תמיד מעל pcStream ו-inc. GMM (מספר שלוש וארבע מימין).

מדד נוסף אותו הראו כותבי המאמר הוא קצב עיבוד החבילות של הנתב שלא נפגע מהרצת המודל. המערכת (כולל הנתב) רצה על Raspberry PI שהיא חומרה יחסית חלשה, ולמרות זאת ביצועי הנתב לא נפגעו.

מהגרף ניתן לראות שקצב עיבוד החבילות עולה כאשר משתמשים ביותר אוטו-אנקודרים בשכבת האנסמבל. הסיבה לכך היא ירידה במספר המאפיינים שימופו לכל אוטו-אנקודר. כפי שתואר לעיל, ממפה המאפיינים מנסה לקבץ מאפיינים לאשכולות לפי מספר האוטו-אנקודרים (אילוץ שעליו לעמוד בו), וככל שמספר זה גדול יותר, פחות מאפיינים ימופו לכל אחד (כדי לעמוד באילוץ) וכתוצאה מכך גודלו של כל אוטו-אנקודר ייקטן. הקטנה של כל אוטו-אנקודר תביא לירידה בכוח העיבוד שכל אחד צורך. לדוגמא, עם k=35 קצב עיבוד החבילות בשלב האימון עומד על 5400 חבילות בשניה ובזמן הריצה עומד על 37300 החבילות בשניה. ביצועים טובים כשלוקחים בחשבון את החומרה עליה המערכת רצה.

סיכום:

המאמר מציג את מערכת Kitsune מערכת ראשונה מסוגה לזיהוי אנומליות. המערכת הייתה הראשונה לשלב Autoencoders בצורה מקוונת כדי לזהות התקפות סייבר ברשת הארגונית. המערכת הינה plug-and-play בכך שהיא אינה צריכה תיוגים כדי ללמוד ומסוגלת לרוץ על נתבים בעלי חומרה חלשה בלי לפגוע בביצועים שלהם דבר שיכול להוביל לפגיעה ברשת כולה.

שיתוף פעולה: הסקירה נכתבה יחד עם עדן יבין.