Technologic Papers

חלק ב' - SELinux - The Practical Way

מאת עדי מליאנקר

הקדמה

SELinux היא module אבטחתי בלינוקס המחלק את המערכת לאובייקטים בעלי Security Context שונים. המערכת מגדירה יחסים מותרים בין האובייקטים השונים על בסיס ה-Context שלהם. SELinux מסתמכת על פוליסות שהן אוסף של modules המכילים חוקים.
תגית נראית בצורה של:

```
type:role:user
```

לעיתים, SELinux מממש מנגנון MCS - הוספת קטגוריות ל-Context שראינו הרגע. עוד מנגנון ש-SELinux עשוי לממש הוא MLS, המאפשר סיווג משאבים עפ"י סיווג MLS. יוסיף דרגה ל-contex. נשים לב שהמשתמש בContext הוא משתמש שהוגדר ב-SELinux. לרוב, אין התאמה חח"ע בין משתמש SELinux למשתמש הלינוקס ועל כן יש לשים לב למיפוי המתבצע בין הקבוצות. דומיין הוא הContext המוקצה לתהליך כלשהוא.

SELinux מאפשר "ירושה" ועל כן אם אין חוק מפורש המציין זאת, אובייקטים חדשים (לדוגמא קבצים, תהליכים), ירשו את ה-Context שלהם מהוריהם. המנגנון עובד במדיניות אישור קפדנית - אם לא התייחסנו ישירות לאיפשור פעולה, הדבר נאסר. באופן דיפולטיבי, CentOS יריץ את ה-SELinux במדיניות targeted, כלומר מתייחס רק למספר תהליכי מערכת (לדוגמא, ב-CentOS 4 קיימים רק 15 תהליכים בעוד וב-CentOS 5 ישנם 200 תהליכים). שאר התהליכים יוגדרו כ-Undefined Domain ולא יושפעו מSELinux.

ב-SELinux, תהליכים יקבלו את ה-Context של הוריהם במקרה של קריאה לפונקציה fork. בקריאה ל-exec, או setcon הוא עשוי לשנות Context. בקבצים, החלק של ה-user מגיע מהתהליך היוצר. החלק של ה-role בדר"כ יהיה בתחילה object_r (כל עוד לא מוגדר Role Transition). החלק של ה-type יהיה בדר"כ כמו אל הנתיב, אלא אם כן מוגדר type transition. ה-range יהיה כמו התהליך היוצר או נמוך ממנו (בתנאי שאין range transition).

Booleans הן מחרוזות המאפשרות לשנות את הפוליסה בזמן ריצה, ללא קימפול חוקים מחדש. להלן כמה דוגמאות:

```
privoxy_connect_any            (on   ,  on)  Allow privoxy to connect any
smartmon_3ware                 (off  ,  off)  Allow smartmon to 3ware
mpd_enable_homedirs            (off  ,  off)  Allow mpd to enable homedirs
xdm_sysadm_login               (off  ,  off)  Allow xdm to sysadm login
xen_use_nfs                    (off  ,  off)  Allow xen to use nfs
```

[thegeekdiary]

הודעות avc (access vector cache) הן הודעות audit שמוכנסות ל-/var/log/audit/auditd.log כש-SELinux דוחה פעולה כלשהיא.

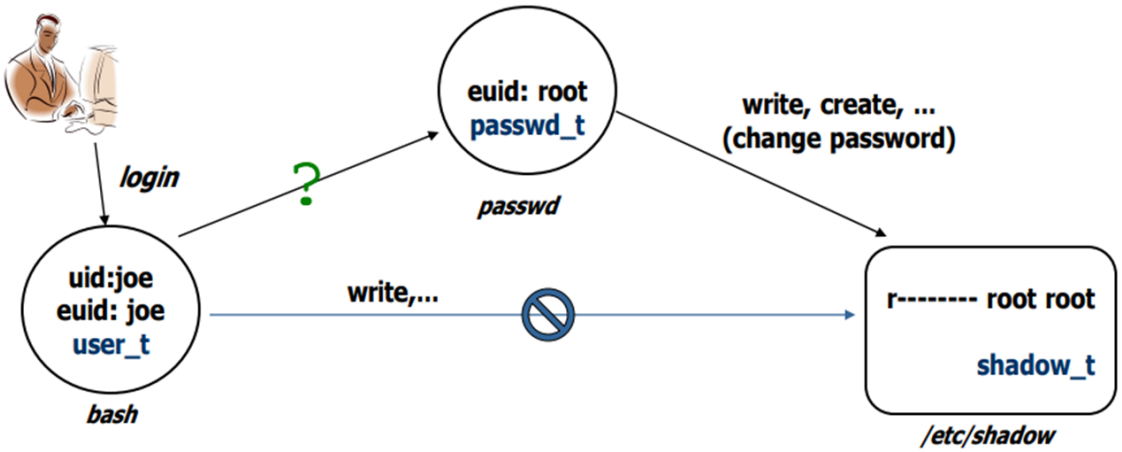
Transitions

Domain transition קורה כאשר תהליך בדומיין אחד מועבר לדומיין אחר ע"י הרצה של תוכנית כלשהיא שמכילה entrypoint type של הדומיין החדש. Entrypoint היא הרשאה הקובעת מי יכול להיכנס לדומיין כלשהוא. נביא דוגמא:

משתמש מעוניין לשנות את סיסמתו ע"י הפקודה passwd. הקובץ /usr/bin/passwd של הפקודה מסומן כ-passwd_exec_t. Passwd נכנס לקובץ /etc/shadow שמתויג כ-shadow_t. חוק דיפולטיבי שקיים ב-SELinux הוא לאפשר לתהליכים שרצים בדומיין passwd_t לקרוא ולכתוב לקבצים בעלי תגית shadow_t. Shadow_t, מצידו מאפשר גישה רק לקבצים שנדרשים לשינויי סיסמאות (לדוגמא, /etc/shadow /etc/gshadow). החוק ב-SELinux קובע שה-entrypoint של הדומיין passwd_t הוא passwd_exec_t. כאשר משתמש מריץ את פקודת passwd, תהליך ה-user shell מבצע המרת דומיין אל passwd_t ויאושר היות וקיים חוק המאפשר זאת.

בדוגמא הזו, ניתן להיכנס לדומיין passwd_t רק בעזרת הרצה של תוכניות המסומנות כ-passwd_exec_t. רק דומיינים מאושרים יכולים להמיר את עצמם ל-passwd_t (לדוגמא תהליך sendmail לא אמור להצליח לעשות זאת כי אין לו סיבה ולכן לא קים חוק המאפשר לו). כמו כן, תהליכים שרצים כ-passwd_t יכולים לקרוא ולכתוב רק לקבצים ספציפיים מאוד (וכך נמנע ניצול של הפקודה כדי לקרוא / לרשום לקבצים אחרים).

להלן הבעיה שהביאה את הרצון ל-domain transition בדוגמא:



[מתוך cse.psu.edu]

נשים לב ש-transition אפשרי רק כאשר:

- יש לדומיין המקורי הרשאות הרצה על הקובץ
- ה-Context של הקובץ נחשב כentry point של דומיין היעד.
- הדומיין המקורי מורשה להיות מומר לדומיין השני.

File transition הוא קונספט שמאפשר לכותב הפוליסה להגיד שאם תהליך a_t יוצר אובייקט ששמו obj בנתיב שמסומן כ-b_t, הוא יקבל Context של c_t. הפעולה מאפשרת להתעלם מהירושה של נתיב כלשהוא, כאשר אובייקט כמו קובץ נוצר בתיקייה.

ישנם סוגים נוספים של transitions שדי דומים ברעיון:

- role transition המאפשר לשנות תפקיד ברגע שתוכנית מורצת
- range transition החלפת range ברגע ההרצה.

פקודות רלוונטיות ב-SELinux

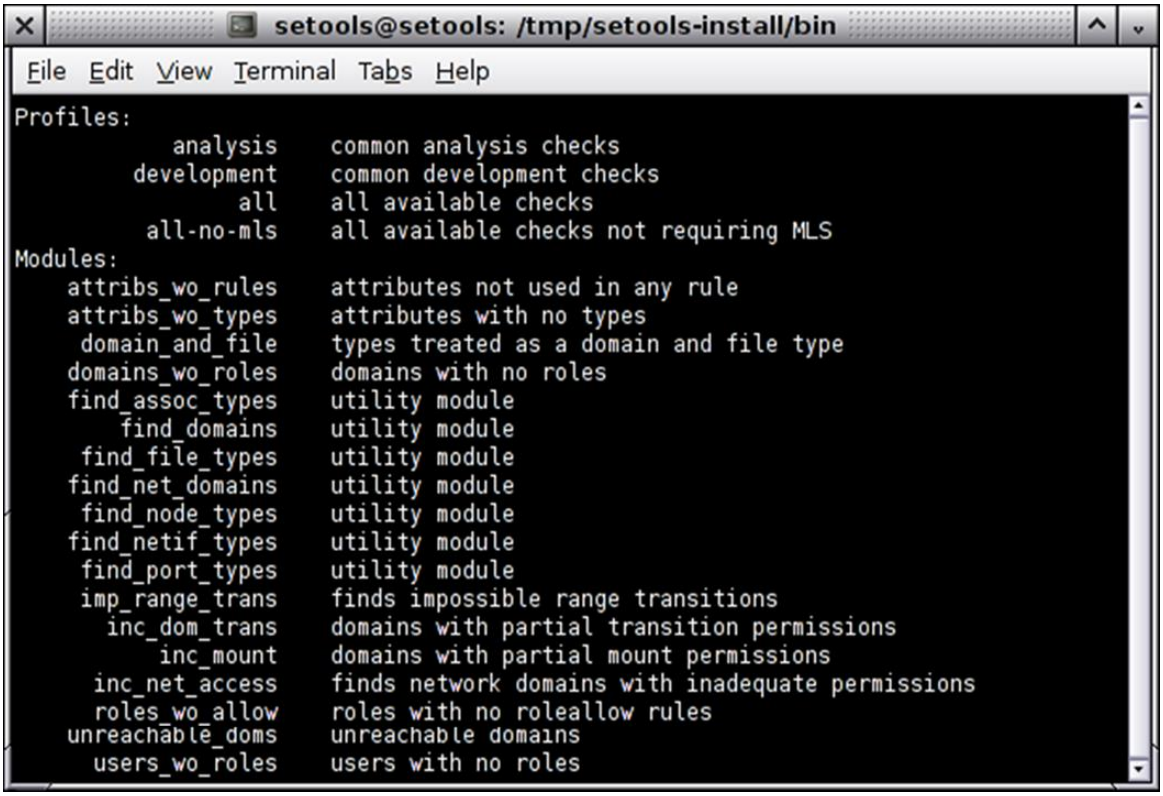
SELinux, ככלי מכיל הרבה פקודות שלכאורה על פי שמם לא היינו יכולים לחברם אל הכלי אך לאחר חיפוש מעמיק ניתן להסיק זאת. במאמר, צירפתי הסבר על רשימה ארוכה של כלים שימושיים בהם מיישמים, בודקי חדירות וחוקרים עשויים להשתמש. עם זאת, הרשמה אינה סופית ועשוייה לכלול עוד פקודות, אך פחות נפוצות. אתם מוזמנים לדלג על חלק זה ולקפוץ ישר לחלק של השימוש הפרקטי בכלים הללו בחלקים הבאים.

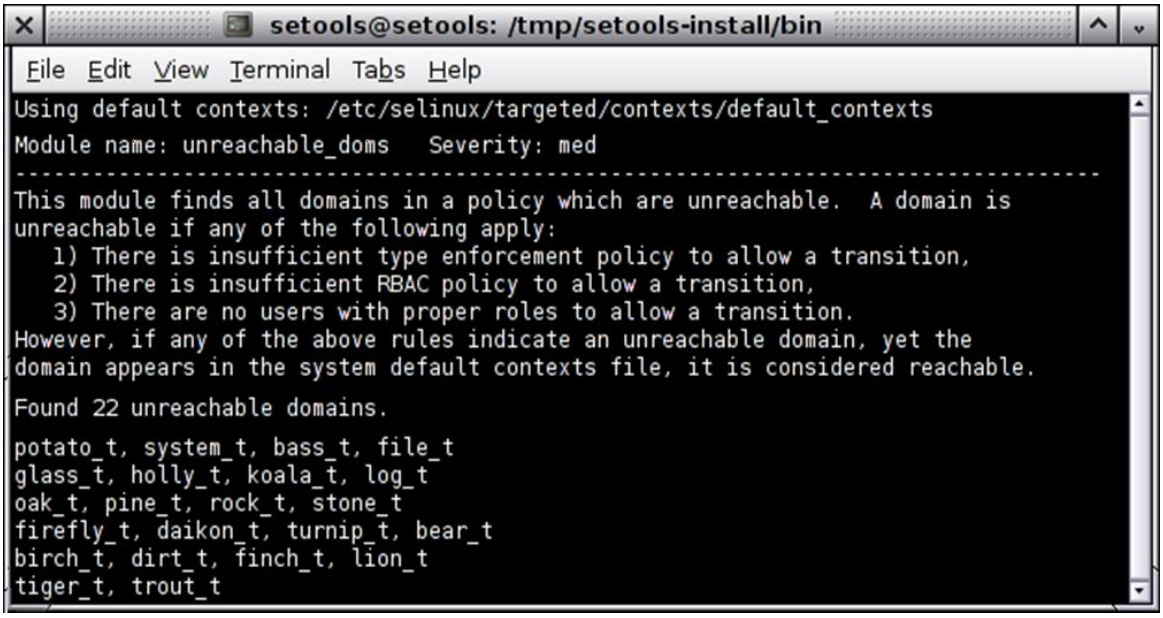
הפקודה הראשונה נקראת **seinfo**. הפקודה, הכתובה בפייטון, מאפשרת למיישם לתחקר את ה SELinux policy, לקבלה בפורמטים שונים ולהוציא ממנה אובייקטים כדוגמת חוקים, משתמשים, Booleans ועוד. **sediff** מאפשרת, כשמה, לחפש שינויים סמנטיים בין 2 פוליסות או בין חלקים ספציפיים (לדוגמא

booleans). הכלי מקטלג שינוי לאחד מ-3 הקבוצות - add, remove, modify . לכלי קיימת גם גרסת gui הנקראת sediffx.

הפקודה **sechecker** מאפשרת לבצע בדיקות מודולריות על פוליסה. היא מכילה מספר רב של מודולים בהם ניתן להשתמש לדוגמא unreachable-doms יתן את כלל הדומיינים בפוליסה.

יש לכלי פרופילים שונים המאפשרים לדו"ח לצאת בהתאם לדרישות המתכנת:





להלן גם דוגמא של פרופיל ובאפשרותנו להוסיף פרופילים משלנו:

```
<sechecker version="1.1">
<profile>
        <module name="find_domains">
                <output value="quiet"/>
                <option name="domain_attribute">
                        <item value="domain"/>
                </option>
        </module>

        <module name="find_file_types">
                <output value="quiet"/>
                <option name="file_type_attribute">
                        <item value="file_type"/>
                </option>
        </module>

        <module name="domain_and_file">
                <output value="short"/>
        </module>
```

[מתוך github.com/TresysTechnology/setools3]

אפשר לראות שהמבנה של פרופיל די פשוט ופשוט מכיל רשימה של מודולים ואת האופן שבו אנו רוצים את ה-output שלהם.

הכלי **apol** הוא כלי gui המאפשר ניתוח פוליסות. הכלי שימושי למערכות רצות וגם כאשר אנו רוצים לבחון את החוקים שלנו, שכתבנו לפני טעינתם למערכת. הכלי בוחן את הקשרים שבין ה-type-ים השונים וכיצד הם משוחחים. כמו כן, הו בודק flow בין שני טיפוסים, לדוגמא אם נרצה שהתוכן של /home (שנחשב כ-user_home_dir) יהיה זמין רק לדומיין httpd_t, apol יחפש בפוליסה את כל הדרכים בflow בין השניים.

sedta מאפשר לבצע אנליזה ל-domain transitions על בסיס אלגוריתמי גרף על מנת לחשב מסלולים בין הטיפוסים השונים. **seaudit** הוא כלי גרפי המאפשר ניתוח לוגים והוצאת מידע של SELinux מלוגים. הכלי בעיקר מוציא מידע של טעינות פוליסות, הודעות avc ושינוי של Booleans. ניתן להשתמש בו גם על מנת לפלטר ולמיין על בסיס שדות מסוימים בהודעה, הכלי ניתן לשימוש גם ב-real time - די שימושי!

indexcon מאפשר ליצור db של security contexes בתוך נתיב מסוים. **findcon** מאפשר לחפש security Context מסוים בתוך מערכת קבצים או db שנוצר ע"י indexcon בעזרת חיפוש מלא / חלקי או regex. הכלי **replcon**, כשמו, מאפשר החלפת תגיות באחרות.

findcon -u user_u .
Find every context whose user is user_u in the current directory and all subdirectories.
findcon -u system_u -t bin_t file_contexts
Find entries user system_u and type bin_t within a file_contexts file, assuming that file_contexts is a file contexts file.

[https://linux.die.net/man/1/findcon]

sesearch מאפשר חיפוש חוקים בתוך פוליסה (לדוגמא חוקים של allow ומתייחסים ליישות x). **chcat** מאפשר הסרת קטגוריה ממשתמש / קובץ - הדבר רלוונטי כאמור כאשר יש לי MCS על המערכת. **restorecon** מאפשר שינוי Security Context של קבצים ל-Context הדיפולטיבי שלהם. כך, אם לדוגמא הקובץ index.html הוא תחת התגית הלא דיפולטיבית של use_home_t, apache לא יוכל לעבוד עם הקובץ בדיפולט ולכן הרצת הכלי תביא להחזרת הרשאות הקובץ ל-http_sys_content_t, איתו apache יכול לעבוד בדיפולט. ניתן בעזרת הכלי גם לראות מה צפוי להשתנות ב-Security Context-ים מבלי לבצע את השינויים בפועל.

sestatus כאמור במאמר הקודם מאפשרת הצגת המצב של SELinux במערכת- האם זה מופעל, היכן נטענה הפוליסה ועל איזה מצב זה נמצא- permissive או enforcing. **chcon מאפשר** שינוי Security Context של אובייקט (לדוגמא משתמש). **runcon** מאפשר למשתמש אחד להריץ פקודה ב-context אחר (ללא שימוש בפקודת sudo) וכל זאת התנאי שיש לו הרשאה להריץ כ-Context אחר. **newrole** יאפשר לנו להריץ shell בעזרת role של Context אחר.

בדוגמא הבאות נשנה את הcontext של httpd.conf (בצורה שווה):

```
chcon system_u:object_r:httpd_config_t:s0 httpd.conf
```

```
chcon -u system_u -r object_r  -t httpd_config_t -l s0 httpd.conf
```

ונעבור קצת לפקודות של Booleans. Getsebool מאפשרת הצגת כלל הבוליאנים (או חלקם) במערכת ואת סטטוס הפעילות שלהם. Setsebool מאפשרת לשנות סטטוס של Booleans. ניתן גם לבצע את השינוי בקובץ: /sys/fs/SELinux/Booleans.

ישנה אפשרות להריץ תוכנה במצב של sandbox ב-SELinux. משמעות הדבר היא שתהיה לתוכנה גישה רק לקבצים שאושרו ספציפית וכן לא תהיה להם גישה רשתית, אלא אם כן צוין אחרת. הפקודה שעושה זאת... כן, ניחשתם נכון, נקראת **sandbox** והיא תעשה האטה יחסית של ריצת התוכנה. חשוב לקחת בחשבון, כאשר משתמשים בכלי, את כל הקבצים שעתידים לשמש את התוכנה. ניקח לדוגמא את הכלי vi, עורך קבצים. כאשר נריץ sandbox על vi בעת יצירת קובץ x, תיווצר בעיה היות ו-vi יוצר גם את הקובץ x.swp וקובץ ששומר את קובץ העריכה ועל כן תיווצר שגיאה.

avcstat מציג סטטיסטיקות על הודעות avc בלוגים. Matchpathcon מאפשר לתשאל קבצי קונפיגורציה על נתיב מסוים ולקבל את ה-default Security Context. SELinux conlist מאפשר הצגה של כלל הcontext-ים הנגישים למשתמש מcontext ספציפי. **audit2allow** מאפשר לקחת את ההודעות בלוגים המעידות על שגיאות גישה ולבנות מהן פוליסה. **auditwhy** הופך הודעות תיעוד להסבר מדוע הגישה נמנעה. **semodule** מאפשר ניהול מודולים של פוליסה (טעינה, הסרה, עדכון, הצגה, כיבוי והפעלה של המודולים). **aursearch** יאפשר לנו לחפש בלוגים על בסיס קטגוריות.

semanage מאפשר הגדרת דברים בפוליסות ללא קימפול או שינוי ה-source לדוגמא מיפוי משתמשי SELinux למשתמשי Linux, הוספת context לקבצים, פורטים מאושרים פר טיפוס, boolean ועוד. כמה מהפעולות המעניינות ב-semanage הן:

- semanage login -l - הצגת מיפוי המשתמשים למשתמשי הלינוקס
- semanage login -a -s <SELinux user> <linux user> - הוספת מיפוי בין משתמש לינוקס ל-SELinux
- semanage user -a -R <role> <SELinux user> - הוספת תפקיד למשתמש SELinux
- semanage port -l - הצגת מיפוי הפורטים לטיפוסים

```
[root@desktop-dgc0qr6 liveuser]# semanage port -l
SELinux Port Type              Proto    Port Number

afs_bos_port_t                 udp      7007
afs_client_port_t              udp      7001
afs_fs_port_t                  tcp      2040
afs_fs_port_t                  udp      7000, 7005
afs_ka_port_t                  udp      7004
afs_pt_port_t                  udp      7002
afs_vl_port_t                  udp      7003
agentx_port_t                  tcp      705
agentx_port_t                  udp      705
ajaxterm_port_t                tcp      8022
amanda_port_t                  tcp      10080-10083
amanda_port_t                  udp      10080-10082
amavisd_recv_port_t            tcp      10024
amavisd_send_port_t            tcp      10025
amqp_port_t                    tcp      5671-5672
amqp_port_t                    udp      5671-5672
aol_port_t                     tcp      5190-5193
aol_port_t                     udp      5190-5193
apcupsd_port_t                 tcp      3551
apcupsd_port_t                 udp      3551
apertus_ldp_port_t             tcp      539
apertus_ldp_port_t             udp      539
asterisk_port_t                tcp      1720
asterisk_port_t                udp      2427, 2727, 4569
audit_port_t                   tcp      60
auth_port_t                    tcp      113
bgp_port_t                     tcp      179, 2605
bgp_port_t                     udp      179, 2605
boinc_client_ctrl_port_t       tcp      1043
boinc_port_t                   tcp      31416
certmaster_port_t              tcp      51235
chronyd_port_t                 udp      323
clamd_port_t                   tcp      3310
clockspeed_port_t              udp      4041
cluster_port_t                 tcp      5149, 40040, 50006-50008
cluster_port_t                 udp      5149, 50006-50008
```

- semanage port -a -t http_port_t -p tcp 822 - הוספת הפורט 822 לפורטים של הטיפוס http_port
- Semanage Boolean - הצגת הbooleanים במערכת ומצבם.

```
[root@desktop-dgc0qr6 etc]# semanage boolean -l
SELinux boolean                 State  Default Description

ftp_home_dir                    (off  ,  off)  ftp_home_dir
smartmon_3ware                  (off  ,  off)  smartmon_3ware
xdm_sysadm_login                (off  ,  off)  xdm_sysadm_login
xen_use_nfs                     (off  ,  off)  xen_use_nfs
mozilla_read_content            (off  ,  off)  mozilla_read_content
ssh_chroot_rw_homedirs          (off  ,  off)  ssh_chroot_rw_homedirs
mount_anyfile                   (on   ,   on)  mount_anyfile
clamscan_read_user_content      (off  ,  off)  clamscan_read_user_content
tftp_anon_write                 (off  ,  off)  tftp_anon_write
zoneminder_anon_write           (off  ,  off)  zoneminder_anon_write
telepathy_connect_all_ports     (off  ,  off)  telepathy_connect_all_ports
spamassassin_can_network        (off  ,  off)  spamassassin_can_network
deny_ptrace                     (off  ,  off)  deny_ptrace
selinuxuser_execmod             (on   ,   on)  selinuxuser_execmod
httpd_can_network_relay         (off  ,  off)  httpd_can_network_relay
openvpn_enable_homedirs         (on   ,   on)  openvpn_enable_homedirs
clamscan_can_scan_system        (off  ,  off)  clamscan_can_scan_system
gpg_agent_env_file              (off  ,  off)  gpg_agent_env_file
telepathy_tcp_connect_generic_network_ports (on   ,   on)  telepathy_tcp_connect_generic_network_ports
sge_domain_can_network_connect (off  ,  off)  sge_domain_can_network_connect
unconfined_mozilla_plugin_transition (on   ,   on)  unconfined_mozilla_plugin_transition
saslauthd_read_shadow           (off  ,  off)  saslauthd_read_shadow
httpd_use_oddjob                (off  ,  off)  httpd_use_oddjob
httpd_can_network_connect_db    (off  ,  off)  httpd_can_network_connect_db
use_ecryptfs_home_dirs          (off  ,  off)  use_ecryptfs_home_dirs
dbadm_manage_user_files         (off  ,  off)  dbadm_manage_user_files
postgresql_can_rsync            (off  ,  off)  postgresql_can_rsync
polipo_connect_all_unreserved   (off  ,  off)  polipo_connect_all_unreserved
httpd_use_gpg                   (off  ,  off)  httpd_use_gpg
```

Ls, netstat, mkdir, id, find, cp,ps הן פעולות הקיימות בכל לינוקס ואינן מתייחסות ל-SELinux. עם זאת, הן מכילות דגל של Z המאפשר ליצור תיקייה/ לחפש/ להעתיק/ לצפות התהליכים ועוד על בסיס ה security contex.

```
[root@desktop-dgc0qr6 etc]# ls -Z
drwxr-xr-x. root root   system_u:object_r:abrt_etc_t:s0  abrt
-rw-r--r--. root root   system_u:object_r:adjtime_t:s0   adjtime
-rw-r--r--. root root   system_u:object_r:etc_aliases_t:s0 aliases
-rw-r-----. root smmsp  system_u:object_r:etc_aliases_t:s0 aliases.db
drwxr-xr-x. root root   system_u:object_r:etc_t:s0       alsa
drwxr-xr-x. root root   system_u:object_r:etc_t:s0       alternatives
-rw-r--r--. root root   system_u:object_r:etc_t:s0       anacrontab
-rw-r--r--. root root   system_u:object_r:etc_t:s0       anthy-conf
-rw-r--r--. root root   system_u:object_r:etc_t:s0       asound.conf
-rw-r--r--. root root   system_u:object_r:etc_t:s0       at.deny
drwxr-xr-x. root root   system_u:object_r:etc_t:s0       at-spi2
drwxr-x---. root root   system_u:object_r:etc_t:s0       audisp
drwxr-x---. root root   system_u:object_r:auditd_etc_t:s0 audit
drwxr-xr-x. root root   system_u:object_r:etc_t:s0       avahi
drwxr-xr-x. root root   system_u:object_r:etc_t:s0       bash_completion.d
-rw-r--r--. root root   system_u:object_r:etc_t:s0       bashrc
drwxr-xr-x. root root   system_u:object_r:etc_t:s0       binfmt.d
drwxr-xr-x. root root   system_u:object_r:bluetooth_conf_t:s0 bluetooth
drwxr-xr-x. root root   system_u:object_r:etc_t:s0       brltty
-rw-r--r--. root root   system_u:object_r:etc_t:s0       brltty.conf
drwxr-xr-x. root root   system_u:object_r:etc_t:s0       chkconfig.d
-rw-r--r--. root root   system_u:object_r:etc_t:s0       chrony.conf
-rw-r-----. root chrony system_u:object_r:chronyd_keys_t:s0 chrony.keys
-rw-r--r--. root root   system_u:object_r:etc_t:s0       colord.conf
drwxr-xr-x. root root   system_u:object_r:etc_t:s0       corosync
```

כתיבת module

אופן כתיבת module הוא תהליך יחסית קליל ובו עלינו ליצור קובץ מסוג te (type enforcement). בדוגמא הבאה, יצרנו קובץ ששמו file.te בתיקייה /usr/share/SELinux/packeges. להלן פוליסה פשוטה:

```
Policy_module(file, 1.0)
 gen_require(`type user_t; type var_log_t;)
Allow user_t var_log_t :dir {getattr search open read } ;
```

בקובץ , השורה Policy_module(file, 1.0) משמעותה שהקובץ file הוא מהגרסא 1.0 (הגרסא תופיע כשנעשה בהמשך semodule -l). שתי השורות שלאחר מכן מכילות את ההצהרה על ה-types בהם נשתמש ולבסוף, מופיע החוק עצמו בשורה האחרונה- איפשור למי שבידו ה-Context של user_t לבצע השגת מאפיינים, קריאה, כתיבה וחיפוש בקבצי var_log_t בעלי role של dir.

לאחר מכן נקמפל את הקובץ בעזרת make -f file.pp. נשים לב ששם הקובץ בסיומת pp. ובקובץ עצמו (בשורה Policy_module(file, 1.0)) צריכים להיות זהים. לבסוף, נטען את הקובץ בעזרת:

```
semodule -I file.pp
```

כעת, אחרי שראינו דוגמא פשוטה, ניכנס יותר לעומק עם קובץ מורכב יותר:

```
policy_module(myapp,1.0)

attribute_role myapp_roles;
roleattribute system_r myapp_roles;

type myapp_t;
type myapp_exec_t;
type myapp_log_t;
type myapp_tmp_t;

role myapp_roles types myapp_t;

domain_type(myapp_t)

domain_entry_file(myapp_t, myapp_exec_t)

logging_log_file(myapp_log_t)

files_tmp_file(myapp_tmp_t)

allow myapp_t myapp_log_t:file append_file_perms;
allow myapp_t myapp_tmp_t:file manage_file_perms;

files_tmp_filetrans(myapp_t,myapp_tmp_t,file)

userdom_use_inherited_user_terminals(myapp_t)
```

[omarine.org]

השורה הראשונה הוסברה קודם לכן. שורות 2-3 מגדירות קבוצת תפקידים שתיקרא myapp_roles ולתוכה נכניס את התפקיד system_r.

ב-4 השורות הנ"ל:

```
type myapp_t;
type myapp_exec_t;
type myapp_log_t;
type myapp_tmp_t;
```

נגדיר 4 טיפוסים, אגלה לכם ש-myapp_t מייצג דומיין, myapp_exec_t מייצג קובץ הרצה, myapp_log_t מייצג קבצי לוגים וmyapp_tmp_t ייצג את קבצי ה-tmp.

```
role myapp_roles types myapp_t;
```

כאן, נקשר את קבוצת התפקידים myapp_roles עם הדומיין myapp_t וכך, הדומיין יקושר לתפקידים שבקבוצה.

```
domain_type(myapp_t)
```

⇔

```
require { attribute domain; }
typeattribute myapp_t domain;
```

כאן, בשורה מצד ימין, נגדיר את myapp_t להיות חלק מקבוצת הדומיינים. הדבר שווה ערך לשורה שמשמאלו.

```
domain_entry_file(myapp_t, myapp_exec_t)
```

```
logging_log_file(myapp_log_t)
```

```
files_tmp_file(myapp_tmp_t)
```

בשלושת השורות האחרונות, קבענו את myapp_tmp_t ,myapp_log_t ,myapp_exec_t להיות ה-Context עבור ה-entrypoint הלוגים וקבצי הtmp בהתאמה.

שני חוקי ה-allow יאפשרו לתהליך שהדומיין שלו הוא myapp_t לשנות את קבצי הtmp והלוגים המקושרים עם הContext שבחוקים.

```
files_tmp_filetrans(myapp_t,myapp_tmp_t,file)
```

בשורה, נבצע file transition עליו דיברנו כבר קודם. נשנה את ה-Context של קבצים שנוצרו ע"י myapp_t בתיקיית ה-tmp ל-myapp_tmp_t.

```
userdom_use_inherited_user_terminals(myapp_t)
```

לבסוף, השורה תאפשר לדומיין myapp_t לקרוא ולכתוב לתוך user ttys, ptys (כלומר התהליך יוכל לדבר עם משתמש דרך הרשאות terminal).

כעת, נוכל לראות את הmodule בין המודולים המותקנים בעזרת semodule -l. ישנם עוד 2 קבצים שניתן להוסיף. הראשון, נקרא interface file שסיומתו if. הקובץ מגדיר את "הפונקציות הציבוריות" של ה-module, אלו שמודולים אחרים יכולים להשתמש בהן והן מוגדרות בתוך קובץ ה-te. השני, קובץ fc מגדיר context של קבצים הקשורים ל-module. שני הקבצים הם בגדר רשות ותוספת לקובץ הte.

ניתן לראות דוגמא לפונקציות שחוקים קיימים מייחצנים בעזרת האתר oss.tresys, לדוגמא:

Interfaces:

`cron_admin_role( role , domain )`

Summary

Role access for admin cron.

Parameters

Parameter:	Description:
role	Role allowed access.
domain	User domain for the role.

`cron_anacron_domtrans_system_job( domain )`

Summary

Execute anacron in the cron system domain.

Parameters

Parameter:	Description:
domain	Domain allowed to transition.

`cron_create_log_files( domain )`

Summary

Create cron log files.

Parameters

Parameter:	Description:
domain	Domain allowed access.

כעת, נסביר קצת על יצירה של קבצי interface.

לרשותנו קובץ ה-te הבא:

```
policy_module(mysysadm, 0.1)
gen_require(`
  type sysadm_t;
')
logging_exec_syslog(sysadm_t)
```

ברצוננו לייחצן את הפונקציה logging_exec_syslog. לשם כך, ניצור קובץ if

```
## <summary>Local adaptation to
the system logging SELinux
policy</summary>

##################################
########
## <summary>
##    Execute the system logging
daemon in the caller domain
## </summary>
## <desc>
##   <p>
##     This does not include a
transition.
##   </p>
## </desc>
## <param name="domain">
##    <summary>
##      Domain allowed access.
##    </summary>
## </param>
#
interface(`logging_exec_syslog',`
  gen_require(`
    type syslogd_exec_t;
  ')
  can_exec($1, syslogd_exec_t)
')
```

[SELinux Cookbook]

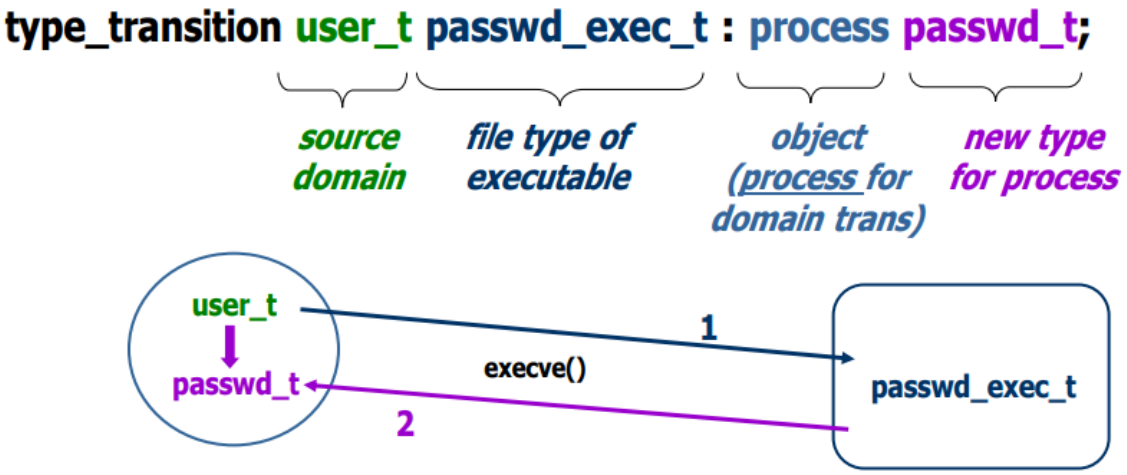
מה שיעניין אותנו בקובץ הוא גוף הפונקציה. משמעות הפונקציה היא שהתהליך שיצויין ב-$1 ע"י תהליך חיצוני שיקרא לפונקציה, יוכל להריץ את הקובץ שהcontext שלו הוא syslogd_exec_t. שם הפונקציה היא logging_exec_syslog.

קובץ fc הוא די פשוט ויראה כך:

```
/bin/myapp  --      gen_context(system_u:object_r:myapp_exec_t,s0)
```

בקובץ, נציין את הpath של הקבצים אליהם אנחנו רוצים להתייחס ביחס לחוק ואת הContext הדיפולטיבי שלהם, זה שאליו נגיע בהרצה של restorcon.

כאשר נרצה לשים חוקים ליצירת domain transition, נצטרך לציין את שם הדומיין ממנו אנחנו רוצים לעבור, entrypoint, התהליך לצורך הdomain transition וסוג התהליך החדש. כמו שניתן לראות כאן:



[cse.psu.edu]

כדי לבצע file transition בפוליסה, נשתמש ב-keyword של:

filetrans_pattern (process contex, dir contex, new object's contex, object name)

ביצוע role transition מתבצע ב-2 שלבים בקובץ ה-te:

- איפשור מעבר תפקידים רק מ-old ל-new בעזרת השורה

```
allow old_r  new_r;
```

- ביצוע ה-role transition רק כאשר ה-entrypoint הוא exec_dir:

```
allow old_r  new_r;
```

לאחר מכן ניתן לבצע domain transition על הדומיין המריץ את הפעולה. דבר נוסף שניתן לבצע בפוליסה הוא לבצע התנייה: אם Boolean מסוים מוגדר, נבצע פעולות כלשהן. נעשה זאת בעזרת ה-template:

```
If ( Boolean ) { <actions> }
```

יצירת אובייקטי SELinux חדשים

כאשר נרצה להוסיף Context חדש למערכת, נשים בקובץ ה-te את השורות:

```
Type <new_tag_name_t>;
Files_type(<new_tag_name_t>);
```

על מנת ליצור Booleans עבור הפוליסה שלנו, נשים בקובץ הte את הפונקציה gen_tunable שתקבל שם של פונקציה וערך דיפולטיבי. בהמשך, יש לציין עבור כל פונקציה מה המשמעות שלה. לדוגמא:

```
gen_tunable(skype_use_audio, false)

tunable_policy(`skype_use_audio',`
  dev_read_sound(skype_t)
  dev_write_sound(skype_t)
)
```

כאן, כאשר נפעיל את ה-boolean שנקרא skype_use_audio, נקרא לפונקציות dev_read_sound ו-dev_write_sound.

יצירת משתמש משתמש חדש (linuxuser) המקושר למשתמש SELinux קיים (user_u) יכול לקרות בעזרת הפקודה הבאה:

```
Useradd -Z user_u linuxuser
```

הוספת תפקידים למשתמש שיצרנו בעזרת הפקודה:

```
User <username> roles {<role 1>  <role 2>};
```

יצירת משתמש SELinux חדש בעזרת semanage:

```
#semanage user -a -L s0 -r "s0- s0:c0.c127" -R user_r finance_u
```

[SELinux cookbook]

על מנת ליצור תפקיד חדש, נבצע את הצעדים הבאים:

- ניצור module על שם התפקיד.
- בפוליסה, נשים את השורה הבאה:

Userdom_login_user_template(<user role>)

- נקצה לתפקיד את ההרשאות המתאימות, לדוגמא עבור postgres admin נשתמש בפונציה postgresql_admin (<domain name>, <new role name>)
- בקובץ /etc/SELinux/mcs/contexts נקצה את התפקיד לדומיין הרלוונטי כערך דיפולטיבי, לדוגמא pgsqladm_r:pgsqladm_t.
- נוסיף את השורה שהוספנו קודם גם בקובץ /etc/SELinux/mcs/contexts על מנת להגיד למערכת לאיזה משתמש לעשות transition בעת הצורך.
- לבסוף, ניתן לוודא את השינויים שהוספנו בעזרת seinfo -r | grep <our new role>

SELinux-ו Containers

container-ים היא יחידת תוכנה המכילה בתוכה גם את כלל המשאבים הדרושים לריצת התוכנה- אם אלו סיפריות, תוכנות משנה ועוד. כך, לא נתקל יותר בבעיות של תיקייה כלשהיא לצורך ריצת התוכנה ונקבל סוג של הפרדה מהתוכנות האחרות. במבט ראשון, אנחנו לא צריכים לראות בעיה בשימוש של container-ים ביחד עם SELinux. עם זאת, מדובר "מערכת עצמאית" החיה בתוך המערכת הקיימת ועשויה גם להתממשק עימה, הן מבחינה רשתית והן מבחינת הזיכרון והקבצים.

Container-ים מקבלים לרוב context של container_t . container-ים יכולים בדר"כ להריץ ולקרוא מתוך usr/ ואת רוב התוכן בתיקיית etc/. הקבצים בתיקייה /var/lib/docker, /var/lib/containers יקבלו את התגית container_var_lib_t. SELinux ימנע לרוב ביצוע mount של קבצים מוגנים אל ה-container. רק קבצים עם context של container_file_t ניתנים לכתיבה ע"י ה-container-ים.כאשר יש במערכת מנגנון mcs, כל container יקבל mcs level על בסיס ה-pid של תהליך ה-docker שיצר את ה-container. כאשר יש בעסק openshift, ה-mcs level יקבע על בסיס ה-uid. גם במקרה שיש ברשותנו מנגנון mls, יהיה שימוש בשדה זה.

על מנת לחסום גישה לתיקיית usr/ מה-container, עלינו לתייגה מחדש. SELinux מורשה לרוץ עם container-ים רק ב-CentOS ו-redhat.

אגב, פחות קשור לנושא, אך איך נראית ההתייחסות למכונות וירטואליות? הבאתי טבלה שתציג את הדבר:

Type	SELinux Context	Description
Virtual Machine Processes	system_u:system_r:svirt_t:MCS1	MCS1 is a randomly selected MCS field. Currently approximately 500,000 labels are supported.
Virtual Machine Image	system_u:object_r:svirt_image_t:MCS1	Only processes labeled *svirt_t* with the same MCS fields are able to read/write these image files and devices.
Virtual Machine Shared Read/Write Content	system_u:object_r:svirt_image_t:s0	All processes labeled *svirt_t* are allowed to write to the svirt_image_t:s0 files and devices.
Virtual Machine Image	system_u:object_r:virt_content_t:s0	System default label used when an image exits. No *svirt_t* virtual processes are allowed to read files/devices with this label.

[redhat.access]

SELinux היא מנגנון מורכב ולכן לא פעם, נצטרך לבצע troubleshoot על מנת להבין מדוע המנגנון לא עובד כהלכה. סיבות שונות שעשויות לגרום לבעיות ב-SELinux הן:

- קובץ עם Context שגוי
- תהליך עם Context אחר
- באג בפוליסה
- ניסיון חדירה למערכת

SELinux מכניס את ההתרעות שלו ל-/var/log/audit/audit.log בעזרת auditd daemon וכאשר הוא אינו רץ, אל הקובץ /var/log/messages. ההודעות מסומנות לרוב עם המילה avc. ישנם כלים כמו sealert היכולים להפוך את הקריאה מהלוגים ל-human readable. באפשרותנו להשתמש בכלים כדוגמת dontaudit בכדי להציג רק הודעות רלוונטיות בלוגים.

להלן הסבר על השגיאות הרלוננטיות שיכולות להופיע בלוגים:

Auditd Record Type	Description
AVC	Messages that are generated by the kernel whenever access to an operation is granted or denied. Audit messages can be generated when access is granted by using **auditallow** policy rules, and messages can also be prevented from being logged when access is denied by using **dontaudit** rules.
USER_AVC	Similar to **AVC** messages, but are generated by userspace programs that use the SELinux security server. These programs are known as userspace object managers, and includes D-Bus and systemd.
SELINUX_ERR	An error emitted by the SELinux security server in cases where an operation fails and an AVC doesn't sufficiently describe the issue. This typically occurs when systemd tries to transition to a service that has NoNewPrivileges=True in it's unit file, but does not have type bounds in the SELinux policy between systemd and its own type. This specific issue can also occur if a process wishes to **setcon(3)** to change the context of one of its threads. It can also occur when a program tries to set an invalid context, e.g., PAM uses **setcon(3)** to change to the login users domain and misconfigured SELinux logins can result in an incorrect context being returned from SELinux userspace.
USER_SELINUX_ERR	Similar to **SELINUX_ERR**, messages of this type are logged when a userspace object manager experiences an SELinux related error.
MAC_POLICY_LOAD, USER_MAC_POLICY_LOAD	A message that is logged when a SELinux policy is loaded. The USER variant is also emitted by userspace object managers to notify that they have processed the policy load.
MAC_CONFIG_CHANGE	A message that is logged when an administrator changes the value of an SELinux boolean using **setsebool**.
MAC_STATUS	When the status of SELinux is changed from enforcing to permissive or vice versa.
USER_ROLE_CHANGE	A message that is logged when a user changes role via the **newrole(1)** command, at login time, or by using **sudo**. Only applicable to Role-Based Access Control.
USER_LABEL_EXPORT	Logged whenever a user exports a labeled object using CUPS.

NOTE: There are several other SELinux related audit events that are used in IPSec/NetLabel that are not covered here at this time. Those events are **MAC_UNLBL_ALLOW**, **MAC_UNLBL_STCADD**, **MAC_UNLBL_STCDEL**, **MAC_MAP_ADD**, **MAC_MAP_DEL**, **MAC_IPSEC_EVENT**, **MAC_CIPSOV4_ADD**, **MAC_CIPSOV4_DEL**.

[wiki.CentOS.org]

דרך אחת לבצע troubleshoot היא ראשית לאשש שפעולה כלשהיא נדחתה בעזרת חיפוש עם ausearch בקבצי לוגים:

```
# ausearch -m AVC,USER_AVC -ts recent
time->Thu Feb 18 14:24:24 2016
type=AVC msg=audit(1455805464.059:137): avc:  denied  { append } for  pid=861 comm="httpd"
name="error_log" dev="sdb1" ino=20747 scontext=system_u:system_r:httpd_t:s0
tcontext=system_u:object_r:var_run_t:s0 tclass=file permissive=0
```

[redhat.access]

ניתן גם להשתמש ב-journalctl (אחד מה-daemon שמבצע תיעוד) על מנת למצוא מידע נוסף:

```
   # journalctl -t setroubleshoot --since=14:20
-- Logs begin at Fri 2016-01-15 01:17:17 UTC, end at Thu 2016-02-18 14:25:21 UTC. --
Feb 18 14:24:24 fedora.23.virt setroubleshoot[866]: SELinux is preventing httpd from append
access on the file error_log. For complete SELinux messages. run sealert
-l e9d8fa2e-3608-4ffa-9e72-31a1b85e460b
```

[redhat.access]

על פי הפלט אתם יכולים לנחש מה יהיה הצעד הבא שנעשה:

```
# sealert -l e9d8fa2e-3608-4ffa-9e72-31a1b85e460b
    SELinux is preventing httpd from open access on the file /var/log/httpd/error_log.

***** Plugin restorecon (99.5 confidence) suggests   ************************

If you want to fix the label.
    /var/log/httpd/error.log default label should be httpd_log_t.
    Then you can run restorecon.
    Do
    # /sbin/restorecon -v /var/log/httpd/error_log
```

[redhat.access]

ואז אפשר לראות שהקובץ /var/log/httpd/error_log קיבל תגית שאינה נכונה ולכן SELinux ידחה אותו.

הפקודות chcon וrestorcon יכולות לסייע לנו בשחזור הרשאות של קבצים ושינוי Context של אובייקטים. כדי לאפשר לתוכנה להאזין בפורט מסוים, ניתן להשתמש ב-semanage port.

לעיתים יהיה לנו קל יותר לעקוב אחרי המתרחש בלוגים בעזרת permissive mode עבור SELinux. לשמחתנו, אנחנו יכולים לבחור אילו דומיינים יהפכו ל-permissive במקום להפוך את כלל SELinux ל-permissive (דוגמא לשימוש היא semanage permissive -a httpd_t שתהפוך רק את הדומיין httpd_t ל-permissive).

כאמור, שגיאות נכתבות אל /var/log/audit/audit.log:

```
type=AVC msg=audit(1226270358.848:238): avc:  denied  { write } for  pid=13349
comm="certwatch" name="cache" dev=dm-0 ino=218171
scontext=system_u:system_r:certwatch_t:s0 tcontext=system_u:object_r:var_t:s0
tclass=dir

type=SYSCALL msg=audit(1226270358.848:238): arch=40000003 syscall=39 success=no
exit=-13 a0=39a2bf a1=3ff a2=3a0354 a3=94703c8 items=0 ppid=13344 pid=13349
auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=
(none) ses=4294967295 comm="certwatch" exe="/usr/bin/certwatch"
subj=system_u:system_r:certwatch_t:s0 key=(null)
```

[https://access.redhat.com]

בתמונה הנ"ל, ניתן לראות ש-certwatch נכשל במלאכת הכתיבה אל האובייקט:

tcontext=system_u:object_r:var_t:s0

נשתמש ב-audit2allow על מנת להבין את סיבת השגיאה:

```
~]# audit2allow -w -a
type=AVC msg=audit(1226270358.848:238): avc:  denied  { write } for  pid=13349
comm="certwatch" name="cache" dev=dm-0 ino=218171
scontext=system_u:system_r:certwatch_t:s0 tcontext=system_u:object_r:var_t:s0
tclass=dir
        Was caused by:
                Missing type enforcement (TE) allow rule.
        You can use audit2allow to generate a loadable module to allow this access.
```

[https://access.redhat.com]

אם כן, הבעיה היא מחסור בחוק מאפשר , אותו גם ניתן ליצור בעזרת audit2allow:

```
~]# audit2allow -a


#============= certwatch_t ==============
allow certwatch_t var_t:dir write;
```

[https://access.redhat.com]

ולבסוף, ניתן להשתמש בכלי גם לצורך יצירה וקימפול החוק שנוצר קודם לכן.

```
~]# audit2allow -a -M mycertwatch
******************** IMPORTANT ***********************
To make this policy package active, execute:

semodule -i mycertwatch.pp

~]# ls
mycertwatch.pp  mycertwatch.te
```

[https://access.redhat.com]

יש לשים לב שלעיתים החוק הנוצר פותח יותר ממה שהוא אמור ועל כן, יש להיזהר מכך.

קבצי SELinux מעניינים

כעת, ניכנס במעט לקבצים מרכזיים ב-SELinux (בנוסף לקבצים שתיארנו במאמר הקודם).

בתיקייה: **/etc/SELinux<filename>/policy/** נוכל למצוא את קובץ הפוליסה הבינארי וקבצי קונפיגורציה.

בתיקייה: **/etc/SELinux<filename>/src/policy/** נמצא את קבצי ה-src של הפוליסות, וב-**/etc/SELinux<filename>/contexes/** נמצא מידע עלcontex-security שקיימים. ב-**/etc/SELinux<filename>/modules/Booleans.active/** נמצא מידע על ה-boolean-ים במערכת ואת קובץ הקונפיגורציות **/etc/SELinux/config**.

בקובץ: **/etc/SELinux/semanage.conf** נוכל לראות הגדרות הנוגעות לפקודה semanage ובקובץ **/etc/SELinux/{SELINUXTYPE}/seusers** נמצא רשימה של משתמשים ב-SELinux.

Setroubleshootd הוא שירות המאפשר איסוף של אירועי audit מהקרנל ומבצע ניתוח שלהם. sealert יעשה לדוגמא שימוש בשירות זה וניתן לקנפגו ב-**/etc/setroublshoot/setroublshoot.cfg**. חוץ מהפעולות read, write וsearch שראינו קודם, ניתן לשים פעולות נוספות בחוקי ה-SELinux וניתן למצוא אותן ב-**/sys/fs/SELinux/class/file/perms**.

יצירת חוק בצורה טובה ומאובטחת

לרוב, נרצה לעשות חוקים לתוכנות שאינן מותקנות דיפולטיבית בלינוקס (אם זה skype, משחק כלשהוא או אפילו תוכנה שלכם). בכל המקרים, אין חוקים ספציפיים לתוכנה שלכם היות ו-SELinux לא מכירה את ההתנהגות הספציפית של התוכנה ולכן נותנת לו חוק גנרי, כמו שאר התוכנות מאותו הסוג. בחלק זה נסביר איך כותבים חוקים לתוכנות כאלו.

ראשית, עלינו לחקור קצת את האפליקציה. עלינו:

- לנסות למפות את הקבצים והנתיבים שהאפליקציה צריכה.
- לחפש על האפליקציה באינטרנט
- לשחק איתה במצב sandbox
- למפות את ההרשאות שהאפליקציה משתמשת בהן.
- למפות את הפורטים בהם היא משתמשת- איפה היא מאזינה ולמי היא מתחברת.
- איך היא מתקשרת עם דומיינים-תוכנות אחרות?
- האם התוכנה צריכה להיכנס ל-kernel או לחומרה?

ניקח כמקרה קצה את תוכנת סקייפ. ראשית, התיקיות בהן הוא משתמש הן- התיקייה בה הוא מכיל את הקונפיגורציות שלו (/skype/~). כמו כן היא תשתמש ב-downloads/ על מנת להעלות ולהוריד משם קבצים. כמו כן יש את כלל הסיפריות בהן התוכנה משתמשת, אך במקרה זה לא נתייחס אליהן היות והן משמשות לרוב את כלל התוכנות. ניתן גם לאתר את הקבצים בעזרת rpm -ql <software>:

```
[root@desktop-dgc0qr6 etc]# rpm -ql rpm
/bin/rpm
/etc/rpm
/usr/bin/rpm2cpio
/usr/bin/rpmdb
/usr/bin/rpmkeys
/usr/bin/rpmquery
/usr/bin/rpmverify
/usr/lib/rpm
/usr/lib/rpm/macros
/usr/lib/rpm/platform
/usr/lib/rpm/platform/alpha-linux
/usr/lib/rpm/platform/alpha-linux/macros
/usr/lib/rpm/platform/alphaev5-linux
/usr/lib/rpm/platform/alphaev5-linux/macros
/usr/lib/rpm/platform/alphaev56-linux
/usr/lib/rpm/platform/alphaev56-linux/macros
/usr/lib/rpm/platform/alphaev6-linux
/usr/lib/rpm/platform/alphaev6-linux/macros
/usr/lib/rpm/platform/alphaev67-linux
/usr/lib/rpm/platform/alphaev67-linux/macros
/usr/lib/rpm/platform/alphapca56-linux
/usr/lib/rpm/platform/alphapca56-linux/macros
/usr/lib/rpm/platform/amd64-linux
/usr/lib/rpm/platform/amd64-linux/macros
/usr/lib/rpm/platform/armv3l-linux
/usr/lib/rpm/platform/armv3l-linux/macros
/usr/lib/rpm/platform/armv4b-linux
/usr/lib/rpm/platform/armv4b-linux/macros
```

את משאבי הרשת נשיג בעזרת הסנפת התווך מספר רב של פעמים או netstat ומציאת דברים משותפים. התוכנה הינה גרפית ולכן היא מתממשקת עם x11 server. ולסיום, התוכנה צריכה גישה למצלמה ולממשקי הקול בחומרה, כמו כן היא צריכה גישה לterminal על מנת להציג שגיאות.

כעת, ניצור קובץ te שיכיל את החוקים. נגדיר 4 משתנים שיכילו את הלוגיקה שראינו קודם- skype_t (הדומיין של התהליך), skype_exec_t (קבצי ההרצה של skype), skype_home_t (קבצי הקונפיגורציות של skype), skype_tmpfs_t (קבצים המתממשקים עם x11 ואחרים). לאחר שיקלול כלל החוקים, נגיע לכתיבת הקובץ הבא (לא אכנס לכלל הפונקציות כאן כי הן לא העיקר בחלק זה, אך ניתן להבין על פי שמן מה קורה בהן באופן כללי):

```
policy_module(myskype, 0.1)
# assigned variables and their types as seen before
attribute_role skype_roles;
type skype_t;
type skype_exec_t;
userdom_user_application_domain(skype_t, skype_exec_t)
role skype_roles types skype_t;  // assigned skype_roles to the domain
skype_t
type skype_home_t;
userdom_user_home_content(skype_home_t)
```

```
type skype_tmpfs_t;
userdom_user_tmpfs_file(skype_tmpfs_t);
# Allow manage rights on ~/.Skype (skype_home_t)
manage_dirs_pattern(skype_t, skype_home_t, skype_home_t)
manage_files_pattern(skype_t, skype_home_t, skype_home_t)
userdom_user_home_dir_filetrans(skype_t, skype_home_t, dir, ".Skype")

# Shared memory (also needed for X11) -allow manage rights of fifo,
sockets,   ## files and symbolic links
manage_files_pattern(skype_t, skype_tmpfs_t, skype_tmpfs_t)
manage_lnk_files_pattern(skype_t, skype_tmpfs_t, skype_tmpfs_t)
manage_fifo_files_pattern(skype_t, skype_tmpfs_t, skype_tmpfs_t)
manage_sock_files_pattern(skype_t, skype_tmpfs_t, skype_tmpfs_t)
## file transition, when skype_t creates a file/fifo or socket in
tmpfs_t, then lable it ## as skype_tmpfs_t
fs_tmpfs_filetrans(skype_t, skype_tmpfs_t, { file lnk_file fifo_file
sock_file })

# Application is an X11 application - giving approach to build
additional types and ## enable x11 support
xserver_user_x_domain_template(skype, skype_t, skype_tmpfs_t)

# Network access
corenet_tcp_bind_generic_node(skype_t)
corenet_udp_bind_generic_node(skype_t)
# Central skype services allow
corenet_tcp_connect_http_port(skype_t)
corenet_tcp_connect_all_unreserved_ports(skype_t)
# Listen for incoming communication
corenet_tcp_bind_all_unreserved_ports(skype_t)
corenet_udp_bind_all_unreserved_ports(skype_t)

# Voice and video calls
dev_read_sound(skype_t)
dev_read_video_dev(skype_t)
dev_write_sound(skype_t)
dev_write_video_dev(skype_t)
# Terminal (tty) output
userdom_use_user_terminals(skype_t)
```

תמיד נוכל גם להסתכל על חוקים מוכנים של תוכנות אחרות בעלות פונקציונליות דומה על מנת לקבל רעיונות (היות וSELinux מתייחס להתנהגות הצפויה של דומיין ולשתי התוכנות התנהגות די דומה).

הצעד הבא, הוא כתיבת קובץ fc שיכיל התיחסות ל-Context של קבצי האפליקציה וקובץ if. לשניהם לא נתייחס כאן היות וראינו תהליך יצירה שלהם קודם לכן. רק אגיד שלפני היצירה של קובץ ה-fc, כדאי לראות מה ה-Context של הקבצים בעזרת ls -Z ולראות שאין להם Context שקובץ fc העניק להם (אם זה המצב, צריך שרק אחד מקבצי ה-fc יתייחס לקבצים).

לבסוף, נשתמש בטכניקות שראינו בחלק של troubleshoot- ausearch וחבריו על מנת למצוא הודעות avc בעייתיות ולטייב את הפוליסה בעת השימוש בה.

לסיכום

במאמר, הצגתי דרכים שונות להשתמש ב-SELinux. כמובן, שמדובר רק בקצה הקרחון והמסמך הרשמי בגודל ~350 עמודים בנושא יכול להעיד על כך. אני חושב שברגע שכלים כאלו עלו על השולחן, החובה עלינו היא להשקיע בהם כמה שיותר על מנת למצות אותם ככל הניתן.

אני מאתגר אתכם, כאנשי אבטחה לא להסתפק רק במאמר הזה, אלא לקרוא ולהעמיק עוד על מנת להבין כיצד פורצים מערכות כאלו מחד ולבנות מערכות הרבה יותר מוגנות ובטוחות אחר כך מאידך. במבט לעתיד, יש module המתחרה עם SELinux שנקרא appguard ואני חושב שכדאי ללמוד גם אותו (למרות שלא יצא לי להיכנס אליו).

ולסיום, תשמרו על עצמכם ותשמרו גם עלינו במרחב הציבורי ובמרחב הקיברנטי ☺.

על המחבר

עדי מליאנקר, 23, חוקר אבטחה במספר תחומים וכן penetration tester .

מייל: temporarynothing0@gmail.com

ביביליוגרפיה

- https://github.com/TresysTechnology/setools3/wiki#sechecker
- https://www.linuxtopia.org/online_books/CentOS_linux_guides/CentOS_SELinux_guide/rhlcommon-section-0104.html
- https://www.thegeekstuff.com/2017/05/restorecon-examples/
- https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html-single/SELinux_users_and_administrators_guide/index
- http://www.cse.psu.edu/~trj1/cse543-f07/slides/03-PolicyConcepts.pdf
- linux.die.net
- https://www.thegeekstuff.com/2017/07/chcon-command-examples/
- https://www.thegeekdiary.com/understanding-SELinux-booleans/
- https://www.linux.com/training-tutorials/run-applications-secure-sandboxes-SELinux/
- https://access.redhat.com/articles/2191331
- https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security-enhanced_linux/sect-security-enhanced_linux-fixing_problems-allowing_access_audit2allow
- https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux_atomic_host/7/html/container_security_guide/docker_SELinux_security_policy
- https://wiki.CentOS.org/HowTos/SELinux#Creating_Custom_SELinux_Policy_Modules_with_audit2allow
- https://serverfault.com/questions/806596/SELinux-how-to-create-a-new-file-type
- https://omarine.org/blog/SELinux-with-omarine-policy-an-in-depth-look-at-the-security-policy-secure-program-with-its-own-domain-part-1/
- http://oss.tresys.com/docs/refpolicy/api/contrib_cron.html
- SELinux cookbook
- SELinux system administration
- Practical linux security practice
- https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security-enhanced_linux/ch07s02
- SELinux notebook