LINUX KEYLOGGER

Jozef Pivarník
Bachelor Degree Programme (3), FIT BUT
E-mail: xpivar00@stud.fit.vutbr.cz

Supervised by: Boris Procházka
E-mail: iprochaz@fit.vutbr.cz

Abstract: This paper collaborates on methods of logging key strokes in Linux kernel in order to reveal secret passwords. Two methods are presented, both of which are useful for specific purpose. While the first one is effective in catching passwords typed in linux terminal environment, the other one intercepts every key stroke and is designed to be working on the lowest level.

Keywords: Linux, kernel, module, function hijacking, keylogger

1 ÚVOD

Vstup z klávesnice patrí medzi najpoužívanejšie metódy interakcie užívateľa s počítačom. V prostrediach s textovým uživateľským rozhraním je to často jediný možný spôsob. Odposluch stlačených kláves, tiež známy pod anglickým pojmom *keylogging* je možné realizovať na viacerých úrovniach. Z pohľadu operačného systému môžeme rozdeliť keyloggery na:

- Keyloggery využívajúce systémové API a bežiace v užívateľskom priestore.
- Keyloggery využívajúce jadrové API a bežiace v jadrovom priestore.

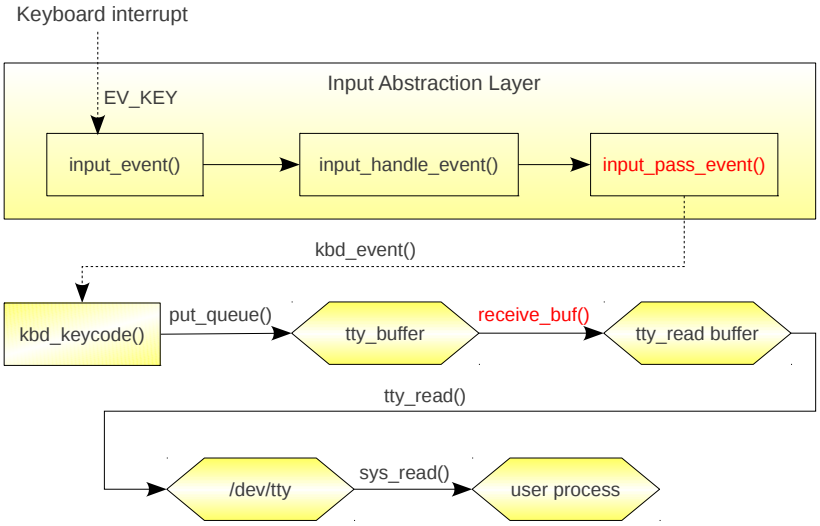
Tento článok sa zaoberá druhou skupinou. Aj napriek tomu, že sú náročnejšie na implementáciu, ich hlavnou výhodou je to, že získavajú väčšie možnosti na ovládanie systému, a preto ich odhalenie nie je jednoduché.

2 VSTUP Z KLÁVESNICE

Každé stlačenie aj uvoľnenie ľubovoľnej klávesy sa klávesnici – ako hardvéru – javí ako samostatná udalosť, ktorá generuje sekvenciu kódov, dĺžky najviac 6 bajtov (najčastejšie však ide o 1 – 2 bajty). Tieto kódy sa nazývajú *scancode*, ktoré sú ďalej prekladané na tzv. *keycode*. Preklad má na starosti ovládač klávesnice a jeho výsledok je posunutý na spracovanie vyšším vrstvám. Ďalej je tento kód prekládaný na *key symbol* pomocou odpovedajúcej mapovacej tabuľky. Výber tabuľky ovplyvňuje kombinácia stlačených modifikátorov (Shift, Control, Alt, AltGr, ShiftL, ShiftR, CtrlL a CtrlR).

Do operačého systému Linux bola od verzie 2.6 zavedená nová vrstva Input Abstraction Layer (IAL), ktorá poskytuje uniformné rozhranie pre spracovanie vstupných udalostí. V prípade, že sa jedná o vstupnú udalosť klávesnice (stlačenie/uvoľnenie klávesy), je ďalej posunuté spracovanie udalosti funkcii `kbd_event()`. Stlačenie klávesy vo výsledku generuje sekvenciu key symbolov, ktoré sú ukladané funkciou `put_queue()` do fronty `tty_buffer`. Periodicky volaná funkcia `receive_buf()` ukladá dáta z tejto fronty do vyrovnávacej pamäte `tty_read buffer`. Pri požiadavku procesu alebo užívateľa na čítanie vstupu je na štandardný vstup volaná funkcia `read()`, ktorá následne volá systémovú funkciu `sys_read()`. V štruktúre `file_operations` je definovaný odkaz na funkciu `read()`, ktorá operuje nad daným typom súboru – v našom prípade je to terminál `/dev/tty`. Systémové volanie `sys_read()` má za následok vyvolanie tejto funkcie, čím

je celý proces ukončený a vstup je predaný procesu, ktorý oň požiadal. Celý proces od vyvolania prerušenia po spracovanie vstupnej udalosti uživateľským procesom je znázornený na obr. 1.



Obr. 1: Proces získania vstupu z klávesnice

3 ODPOSLUCH STLAČENÝCH KLÁVES

Ako bolo uvedené, získanie znaku zo stlačenej klávesy je komplexný proces prechádzajúci viacerými štádiami, ktoré môžu byť prerušené alebo ich činnosť môže byť pozmenená. Za účelom odchytenia stlačených kláves je možné využiť metódu únosu funkcie. Ide o techniku, pri ktorej namiesto pôvodnej funkcie podvrhneme systému našu funkciu, v ktorej tele je možné vykonať takmer akúkoľvek činnosť. Aby ale nedošlo k narušeniu činnosti systému, tak za účelom zachovania pôvodnej funkcionality navyše zavoláme pôvodnú funkciu. Kandidátmi na únos sú funkcie zobrazené na obr. 1.

3.1 ODPOSLUCH NA ÚROVNI TERMINÁLU

Každý terminál je reprezentovaný štruktúrou `tty_struct`. Pre naše účely stačí získať referenciu na túto štruktúru pre každý terminál, ktorý chceme sledovať a nahradiť funkciu `receive_buf()` vlastnou. V prípade, že chceme sledovať všetky terminály, ako výhodnejšie sa javí použiť súbor `/dev/tty`, ktorý reprezentuje práve aktívny terminál. V prípade odstránenia keyloggeru zo systému je potrebné obnoviť odkaz na pôvodnú funckiu `receive_buf()`. Preto je nutné uchovať si ukazateľ na ňu.

Odposluch terminálu môžeme s výhodou využiť na odchytávanie hesiel terminálových aplikácií `sudo`, `su`, `ssh`, `ftp`, `telnet` apod. Pri zadávaní citlivých informácií ako napríklad hesiel dôjde k vypnutiu terminálového echa, čo znamená, že zadávané znaky nie sú tlačené na výstup ani vo forme zástupných symbolov ako napríklad '`*`'. Tento stav je detekovateľný kontrolou lokálnych príznakov príslušnej štruktúry `tty_struct`, konkrétne ide o bity `ICANON` a `ECHO`. Jadro poskytuje makrá `L_ICANON()` a `L_ECHO()`, pomocou ktorých je možné testovať odpovedajúce bity. Pri vypnutí echa nadobudne nasledujúca podmienka hodnotu `true`

```
L_ICANON(tty) && !L_ECHO(tty)
```

3.2 ODPOSLUCH NA NIŽŠEJ ÚROVNI

Nevýhodou vyššie uvedenej metódy odposluchu je, že neodchytáva stlačené klávesy v prostredí inom ako je terminál, resp. pseudoterminál. Preto bol navrhnutý unikátny spôsob odposluchu stlačených

kláves s využitím únosu funkcie jadra podľa [2], ktorý zvláda odposluch každého stlačenia klávesy, teda aj v prostredí X Window System.

Vzhľadom na to, že linuxové jadro definuje umiestnenie vlastných funkcií v pamäti pri kompilácií, dáva tým možnosť zistiť ktorúkoľvek z nich a tú potom uniesť. Zoznam exportovaných symbolov spolu s adresami v pamäti na ktorých sa nachádzajú je uložený v súbore `/proc/kallsyms` alebo `/boot/System.map`. Postup pri unášaní funkcie je nasledujúci:

1. Uloženie prvých 7 bajtov unášanej funckie.
2. Získanie adresy nahradzujúcej unášanú funkciu (jej adresa je dynamická).
3. Nastavenie prvých 7 bajtov unášanej funkcie s ohľadom na takto získanú adresu.

Dôvod, prečo práve 7 bajtov a čo tieto bajty predstavujú je ten, že na uskutočnenie nepriameho skoku sú potrebné na architektúre x86 2 inštrukcie, ktoré majú spolu dĺžku 7 bajtov.

```
static char hf_jmp[7] =
   "\xb8\x00\x00\x00\x00"   // movl  $0,%eax
   "\xff\xe0"               // jmp   *%eax
;
```

Z toho plynie hlavná nevýhoda tejto metódy – neprenositeľnosť. Únos funkcie spočíva v nahradení prológu funkcie (7 bajtov) nepriamym skokom na únosovú funkciu. Tento kód by spôsobil skok na adresu 0, takže ho potrebujeme modifikovať tak, aby na tomto mieste bola adresa nami definovanej funkcie. Pri volaní pôvodnej funkcie potom musí byť najprv obnovených prvých 7 bajtov, aby nedošlo k zacykleniu a po návrate z nej je potrebné ich opäť nahradiť vyššie uvedeným kódom.

Pri implementácii keyloggeru na tejto úrovni bola unesená funkcia `input_pass_event()`. Prepisovanie prológu funkcie a jeho obnova sú kritickou sekciou, ktorá musí byť chránená pomocou zámku. V kontexte funkcie `input_pass_event()` tento problém nie je potrebné riešiť, pretože tá je už pod zámkom volaná. Zo všetkých vstupných udalostí sú pre účely odposluchu stlačených kláves podstatné len udalosti typu `EV_KEY`, teda udalosti generované pri stlačení klávesy.

4 ZÁVER

Obe z uvedených metód odposluchu stlačených kláves boli implementované formou samostatného zásuvného modulu linuxového jadra. Vzhľadom na to, že odposluch je realizovaný na úrovni jadra operačného systému, bežný užívateľ nemá žiadnu možnosť vyhnúť sa mu. Navyše, modul po svojom zavedení do pamäte odstráni referenciu na samého seba zo zoznamu všetkých modulov v systéme, čím komplikuje jeho odhalenie a zároveň sťažuje jeho odstránenie zo systému.

LITERATÚRA

[1] Bovet P. D., Marco C.: Understanding the Linux Kernel. USA: O'Reilly, 2005. ISBN 0-596-00565-2.

[2] Cesare, S.: Kernel function hijacking. November 1999. Dokument dostupný na URL http://biblio.l0t3k.net/kernel/en/kernel-hijack.txt (marec 2011).

[3] Love, R.: Linux Kernel Development. USA: Novell Press, 2005. ISBN 0-672-32720-1.

[4] Rd: Writing Linux Kernel Keylogger. Jún 2002. Dokument dostupný na URL http://www.phrack.com/issues.html?issue=59&id=14, (marec 2011).