Verificarea Monocypher cu FramaC/Eva

Adrian Manea

Cuprins

DE ADĂUGAT SAU CLARIFICAT

INTRODUCERE

Proiectul prezentat este `Monocypher` (LoupVaillant (2019a)), care este o bibliotecă criptografică scrisă de Loup Vaillant, începînd cu 2016. După cum menționează autorul pe pagina LoupVaillant (2017), intenția a fost de a concepe o bibliotecă de criptare suficient de sigură, de rapidă, dar și de simplă pentru a fi folosită atît de el, cît și de alții. Autorul a considerat că bibliotecile actuale ori nu sînt suficient de sigure, ori sînt supraîncărcate pentru necesitățile sale, așa că a pornit în a-și concepe propria soluție. Proiectul este scris în C.

Verificarea proiectului `Monocypher` se va face folosind pachetul `FramaC` (Baudin (2019)), specific pentru limbajul C. Cum `FramaC` este, de fapt, mai curînd un mediu în care se pot rula mai multe unelte de verificare formală, am ales verificarea `Monocypher` cu `Eva`.

Detaliile despre modul de funcționare a proiectului `Monocypher`, cît și a verificării prin `Eva` sînt date în secțiunile următoare.

Toate exemplele și capturile proprii au fost realizate pe o mașină Core i7 8th gen și:

- `frama-c Chlorine 20180502`, instalat via `opam`;
- `Eva 18 Argon`;
- sistemul de operare Manjaro 18 (Illyria), Community Edition i3 (Manjaro (2019));
- editorul text Emacs 26.2 Emacs (2019) cu `acsl-mode` pentru adnotări ACSL (FramaC (2016));
- emulatorul de terminal `st` (Suckless (2019)), cu modificările lui Luke Smith (Smith (2019)).

bucăți de cod și comenzi centrate, pe linie separată!

1 UNELTELE DE VERIFICARE

1.1 FramaC

FramaC este o unealtă de verificare formală dezvoltată de un grup de cercetători de la Commisariat á l'Énergie Atomique (CEA-list) și INRIA din Franța.

În esență, FramaC verifică programe scrise în C, dar este, de fapt, un cadru în care se pot include și utiliza unelte de verificare cu scopuri precise. În varianta cea mai simplă, FramaC folosește un dialect al limbajului C (de fapt, o variantă simplificată numită *C Intermediate Language*) și oferă metode de verificare similare cu uneltele bazate pe *adnotarea codului*, precum Dafny și altele. Limbajul se numește ACSL și este descris detaliat la FramaC (2019a).

FramaC însuși a fost implementat în limbajul OCaml, astfel că poate fi instalat atît individual, cît și prin managerul de pachete opam.

Odată instalat, FramaC oferă două moduri de interacțiune: la nivel de linie de comandă (CLI) sau prin interfață grafică (GUI). Este de menționat faptul că verificatorul nu oferă un editor, astfel că modalitatea recomandată de utilizare este:

(1) se scrie programul C cu editorul preferat;

(2) se rulează din linia de comandă pe fișierele-sursă de verificat;

(3) fie se redirecționează rezultatul într-un fișier separat, de exemplu `frama-c *.c » log`, fie se salvează într-un format binar, specific, cu comanda `frama-c *.c -save mysession.sav`;

(4) se consultă fișierul `log`, dacă s-a folosit prima variantă, care este un fișier text simplu, sau, preferabil, se încarcă formatul salvat specific în interfața grafică, cu comanda `frama-c-gui -load mysession.sav`.

POZE CLI

Dacă se alege varianta simplă, se poate observa că informațiile din `log` nu sînt foarte explicite mereu, astfel că trebuie să știm să interpretăm rezultatele. În general, ele sînt afișate sub forma `[nivel] fișier rezultat`, unde:

- `[nivel]` este nivelul la care se face verificarea (e.g. nivelul de bază, adică direct codul sursă, dacă s-au folosit adnotări și nu se apelează vreun plugin;
- `fișier` este fișierul verificat, despre care se raportează rezultatele;
- `rezultat` este diagnosticul sau concluzia la care a ajuns analizatorul, la nivelul `[nivel]` asupra fișierului `fișier`.

POZĂ log

În varianta cu interfață grafică, însă, putem vedea mult mai multe informații, într-o variantă mult mai flexibilă:

- În panoul din stînga se pot vedea fișierele încărcate spre analizare, care au fost „desfăcute“ în funcțiile conținute, care pot fi analizate individual;
- Panoul median arată modul în care `FramaC` a interpretat codul-sursă, eventual scriindu-și singur adnotări;
- Panoul din dreapta arată codul-sursă încărcat (în care *nu* se poate scrie);
- Panoul din stînga-jos arată uneltele folosite pentru verificare, eventual plugin-urile încărcate;
- Panoul de jos arată mesajele și erorile generate, cu cîmpurile populate în funcție de uneltele de verificare folosite.

Deși poate fi utilizat în această formă, este recomandabil ca `FramaC` să fie apelat folosind plugin-uri specifice. În verificarea proiectului `Monocypher`, vom folosi plugin-ul `Eva` (varianta actualizată [*Evolved*] a *Value Analysis*), care este descris în secțiunea următoare.

1.2 `Eva`

1.2.1 Motivație

Ca majoritatea uneltelor criptografice, `Monocypher` se bazează pe expresii și operații numerice, eventual cu valori foarte mari și prelucrări pe biți. Aceasta îl face să fie posibil vulnerabil atît la operații matematice interzise (precum împărțirea la 0 sau depășirea intervalului de valori permise de tipul de date folosit) sau la scurgeri de memorie.

`Eva` se bazează pe o teorie matematică numită *interpretare abstractă* (Wikipedia (2019)), conform și FramaC (2019b), care îl face în special potrivit pentru verificarea programelor cu conținut aritmetic bogat. De asemenea, `Eva` poate detecta și scurgeri de memorie sau operații nepermise la nivel de memorie alocată, dar pentru verificări mai avansate în special asupra memoriei, este recomandabil să se folosească `Valgrind` (Valgrind (2019)).

2 VERIFICAREA

3 CONCLUZII

BIBLIOGRAFIE

Baudin, P. h. (2019). `FramaC`. site oficial. online, accesat aprilie-mai 2019.

Emacs, E. (2019). Gnu emacs. site oficial. online, accesat aprilie-mai 2019.

FramaC, E. (2016). Emacs `acsl-mode`. GitHub. online, accesat aprilie-mai 2019.

FramaC, G. (2019a). Acsl. site oficial. online, accesat aprilie-mai 2019.

FramaC, G. (2019b). `Eva`. manual oficial. online, accesat aprilie-mai 2019.

LoupVaillant (2017). How i implemented my own crypto. eseu online. online, accesat aprilie-mai 2019.

LoupVaillant (2019a). `Monocypher`. GitHub. online, accesat aprilie-mai 2019.

LoupVaillant (2019b). `Monocypher`. site oficial. online, accesat aprilie-mai 2019.

Manjaro, E. (2019). Manjaro community edition i3. site oficial. online, accesat aprilie-mai 2019.

Smith, L. (2019). `st` — simple terminal (fork). GitHub. online, accesat aprilie-mai 2019.

Suckless, E. (2019). `st` — simple terminal. site oficial. online, accesat aprilie-mai 2019.

Valgrind, E. (2019). Valgrind. site oficial. online, accesat aprilie-mai 2019.

Wikipedia (2019). Abstract interpretation. link. online, accesat aprilie-mai 2019.