Heuristic Methods for Detecting Computer Viruses

Claudiu CONSTANTINESCU
Project Manager – Kepler-Rominfo România

The recent years have shown a very strong increase in the number of malicious code uprising and infections. Almost every time, a new virus or other sort of treat is followed by a wave of similar malicious code that only changes a bit of the original. Also, the new viruses tend to use more and more polymorphism making it impossible to track by string scanning. Those are the main reasons why heuristic scanning has become the only effective method used by modern anti-virus engines.
Keywords*: heuristic scanning, malicious code detection, detecting treats, and computer viruses.*

Căutarea euristică a virușilor a fost introdusă de către Fridrik Skulason în 1989. Metoda constă în identificarea codului virușilor prin analiza comportării specifice a acestora. Căutarea euristică înseamnă analiză de cod program, fără execuția acestuia și identificarea funcțiilor periculoase ce caracterizează un posibil virus. Validarea integrității fișierelor scoate la iveală infecția, programele de detectare identifică virusul prin secvența de identificare pe când programele euristice caracterizează comportamentul codului ca fiind virus. Dezavantajul metodei constă în numărul mare al alarmelor false și al infecțiilor neidentificate.

Majoritatea căutărilor euristice urmăresc de fapt verificarea existenței unor șiruri de octeți căutând uneori numai în anumite regiuni ale fișierelor executabile, cunoscute a fi cele unde se inserează virușii. Motoarele de scanare recente au chiar posibilitatea de a căuta un șir în care sunt inserate caractere wildcard. Aceasta înseamnă ca pot detecta chiar și virușii ușor polimorfici (în sensul de polimorfici mai slabi). Șirurile de octeți căutate de verificările euristice reprezintă secvențe de instrucțiuni sau comenzi potențial malițioase, specifice acțiunilor de infectare și distrugere, secvențe care trebuie însă alese astfel încât sa nu fie comune și altor tipuri de programe.

Aceste verificări implicate de metodele euristice se fac folosind sisteme bazate pe ponderi și/sau sisteme bazate pe reguli (ambele vor fi descrise în continuare în acest articol). Un motor euristic bazat pe un sistem de greutăți (weight-based) acordă fiecărei funcționalități care este detectată în corpul executabilului o anumită greutate în conformitate cu gradul de pericol pe care această funcționalitate îl presupune. Dacă suma tuturor greutăților marcate depășește o anumită valoare o alarmă poate fi declanșată. Aceasta abordare este veche și este practic modul de lucru al primelor motoare euristice. Aproape toate motoarele euristice care se utilizează începând din anul 2002 implementează sisteme bazate pe reguli. Aceasta presupune ca acea componenta a motorului polimorfic ce conduce analiza (numită analizor) extrage anumite reguli dintr-un fișier analizat ce vor fi ulterior comparate cu un set de reguli considerate ca definesc codul malițios. Daca vor fi detectate reguli care se potrivesc se poate declanșa o alarmă.

În prezent există motoare euristice construite pentru fiecare clasă mare de viruși (chiar și pentru cei mai vechi dintre aceștia pe care nu îi mai găsim în realitate pe nicăieri). În decursul anilor, odată cu creșterea complexității virușilor, a crescut și complexitatea motoarelor euristice care au devenit din ce în ce mai sofisticate și inteligente. Primii antiviruși euristici executau căutări simple de șiruri sau șabloane fapt pentru care mai erau numite și «minimized scan string heuristics». Spre exemplu, se caută un șir care efectua atribuirea unei anumite valori «X» unei variabile. O mulțime de motoare euristice specializate pe detectarea virușilor de macro foloseau exact șirul respectiv ca amprentă ce era căutată.

Modalitatea facilă şi evidentă de păcălire a unei verificări cu un astfel de antivirus a fost bineînțeles o reprezentare diferită a valorii atribuite, «X» (de exemplu «Y/2» unde Y este egal cu 2X).
Aceste tehnici de păcălire a motoarelor de scanare bazate pe metode euristice, folosite de programatorii de viruşi, au fost de altfel numite în literatura de specialitate tehnici anti-euristice. În consecință motoarele euristice ce au fost dezvoltate în perioada următoare au fost forțate să caute mai precis şi să analizeze mult mai atent expresiile întâlnite în bucățile de cod pe care le analizau.

Motoarele euristice şi viruşii criptați

Istoric, primele motoare euristice puteau analiza doar ceea ce era vizibil pentru ele; în consecință, viruşii criptați le puneau probleme majore. Ca răspuns la această problemă, motoarele euristice moderne, analizează codul încercând identificarea buclelor de decriptare şi spargerea acestora. Apoi este luată în considerare existența unei bucle de decriptare în contextul tuturor celorlalte funcționalități care mai sunt detectate în respectivul cod.
Pentru a detecta prezenta unei bucle de criptare un antivirus se bazează pe câteva informații concrete cunoscute despre acest gen de rutine. Astfel, prezenta oricărei combinații a următoarelor condiții/instrucțiuni poate indica o bucla de criptare:

- inițializarea unui pointer cu o adresă validă de memorie;
- inițializarea unui contor;
- operații de citire a memoriei în funcție de un pointer;
- operații logice efectuate pe rezultatul obținut din citirea memoriei;
- operații de scriere în memorie a rezultatului operațiilor logice de la punctul anterior;
- manipulări ale variabilei contor;
- executare de operații diverse în funcție de valoarea variabilei contor.

Aceste semnalmente au fost observate prin studierea unui număr mare de viruşi care foloseau mecanisme de criptare. În multe cazuri viruşii încearcă să ascundă această buclă de criptare prin inserarea de instrucțiuni fantomă şi lungirea cât mai mult a buclei pentru a face componenta de analiză a motorului euristic să se piardă în amănunte.
Dacă analizăm metodele de căutare vom vedea că, în majoritatea cazurilor, detectarea buclei de criptare nu este destul de precisă pentru a putea face o clasare exactă a gazdei analizate şi asta mai ales deoarece mulți viruşi utilizează aceleaşi proceduri de criptare sau unele asemănătoare. În lumea viruşilor binari sunt o mulțime de motoare puternice de criptare (de exemplu foarte folositele TPE sau PME) pe când în cazul viruşilor de macro, procedurile de criptare nu sunt folosite prea des.
În scopul detectării şi curățării viruşilor, motoarele euristice deseori folosesc componente de emulare. Aceste sisteme au, printre altele, abilitatea de a sparge şi emula rutinele de criptare. După ce criptarea a fost spartă (a fost atins sfârşitul buclei de criptare), analiza euristică poate începe pe bucata de cod ce tocmai a fost astfel decriptată.
Aplicațiile Visual Basic şi scripturile VB Script sunt exemple tipice de medii complexe în care emulatoarele pot fi foarte folositoare în spargerea criptărilor; totuşi, o emulare completă este deosebit de complexă. În cele mai multe cazuri (cum ar fi spre exemplu familia de viruşi W97M/AntiSocial care utilizează criptare), un număr mare de instrucțiuni criptate şi existența unor macrouri tipice sunt considerate argumente suficiente pentru a detecta aceasta clasa de viruşi macro fără a mai folosi emulatoare.
Partea criptată a unui virus de macro este de obicei stocata în liniile documentului infectat, de unde este prelucrată în interiorul buclei de decriptare. Odată detectate liniile accesate în cadrul acestei bucle se pot face anumite analize şi direct pe aceste linii pentru a determina dacă prezintă sau nu semnalmentele unei bucăți de cod criptat sau este text normal din cadrul documentului. Pentru a lua o astfel de decizie analizorul se bazează pe următoarele semnalmente:

- şirul este suspect de lung (de exemplu are peste 50 de caractere) şi totuşi nu conține de loc spații sau acestea sunt foarte puține (se

poate ține seama şi de o medie statistică a apariției unui spațiu în text normal);

- multe propoziții încep în mod bizar cu cifre;
- şirul conține amestec de cifre cu caractere alfanumerice şi caractere speciale ceea ce nu se întâlneşte în mod normal în componența unui cuvânt.

Dacă am privi textul respectiv cu ochiul liber deja ar fi evident faptul că acolo se execută anumite operații suspecte, ceea ce deja trebuie să fie un motiv suficient pentru un motor euristic pentru a da o alarmă.

În ultimii ani întâlnim motoare care amestecă abilitățile de detectare euristice cu abordări de detectare generică. Acest mod de lucru înseamnă că motorul euristic încearcă să detecteze că un anumit set de funcționalități găsite într-o gazdă aparțin unui virus, partea generică încercând pe baza setului respectiv să categorisească acest virus ca făcând parte dintr-o familie/clasă de cod malițios. Dacă acest lucru reuşeşte câştigurile sunt majore deoarece un virus cunoscut poate fi curățat din fişierul gazdă iar acesta poate fi folosit în continuare pe când dacă ştim doar ca fişierul este infectat dar nu ştim cum arată corpul virusului nu putem decât sa ştergem fişierul respectiv.

Componentele unui motor euristic

În funcție de mediu şi nivelul tehnologic, următoarele componente pot fi găsite în motoarele euristice: emulator de variabile şi de memorie; parser; analizor de flux; dezasamblor şi emulator; sistem bazat pe greutăți şi/sau sistem bazat pe reguli.

La momentul verificării unei bucăți de cod malițios de tip script, primul pas pe care îl face un motor de detectare este normalizarea fişierului de intrare. Acesta înseamnă eliminarea formatării, scurtarea numelor lungi de variabile, împărțirea în entități (token-uri) a şirului, etc.

Odată determinat tipul de fişier, motorul euristic va căuta punctul de intrare al acestuia. În cazul fişierelor binare, această etapă este simplă, majoritatea acestui tip de fişiere având un punct de intrare identificabil. Pentru codul malițios de tip script este posibil să avem mai multe puncte de intrare. Cea mai simplă abordare însă este scanarea completă a programului ignorând fluxul acestuia şi deci, punctele de intrare. Această abordare pierde însă multe secvențe de flux, cum ar fi cele care folosesc pasarea unor parametrii între macro-uri sau funcții. În consecință, deşi este mai simplu să abordăm astfel problema, riscul de a pierde părți esențiale din funcționalitatea implementată este mai mare.

Bucla principală a fiecărui motor euristic trebuie să selecteze şi sa extragă informații (de obicei opcodes ale următoarei instrucțiuni sau ale următoarei linii în cazul scripturilor) şi să le paseze la rutina analizor. Aceasta rutină analizor trebuie să identifice operația şi să poziționeze diverşi indicatori conform identificării efectuate. Tot analizorul inițiază comunicația cu emulatorul de variabile şi emulatorul de memorie.

Clasificarea funcționalității detectate

După ce întregul program a fost analizat, funcționalitatea găsită în acesta poate fi clasată. Această operație este efectuată de sistemul bazat pe greutăți sau sistemul bazat pe reguli.

După cum am menționat mai devreme, sistemul bazat pe greutăți care era foarte folosit la primele motoare euristice acordă fiecărei funcționalități găsite o greutate şi apoi adună toate aceste greutăți. Această tehnologie nu mai este folosita, cel puțin în forma ei de bază, deoarece cauzează foarte multe alarme false clasând greşit fişiere ca suspecte. Pentru viruşi macro această greşeală apare foarte des dacă sunt găsite un număr mare de operații de copiere din documentul curent în template-ul global chiar dacă nici o alta operație potențial malițioasă nu mai este detectată.

Având în vedere comportamentul vechilor sisteme de scanare euristică a apărut nevoia implementării unui sistem care să producă alarme numai dacă erau îndeplinite un set de condiții speciale. Astfel s-a născut ideea sistemelor euristice bazate pe reguli care pot atinge rezultate mult mai bune. Acestea compară funcționalitățile găsite în codul analizat cu setul de reguli iar dacă o regulă

predefinită este găsită în cod este întors un rezultat pozitiv. În funcție de exactitatea potrivirii cu întregul sistem de reguli pot fi obținute rezultate de genul « virus generic » sau « variantă a virusului ABC ». În orice caz însă orice tip de metodă euristică poate întoarce rezultate eronate. În cazul euristicelor bazate pe reguli acest lucru se întâmplă când sistemul de reguli este prost definit.

Concluzii

După ce am făcut o descriere de ansamblu a abordării folosite şi a componentelor motoarelor euristice ne punem problema cât sunt acestea de folositoare pentru utilizatori cât şi pentru companiile din domeniul antivirus. În ultimii ani (2001 - 2004) au existat câteva momente de explozie efectivă a infectării cu viruşi cum au fost Melissa, VBS/Loveletter, Nimda sau BugBear. După fiecare infecție de acest tip s-a putut observa o mare cantitate de cod malițios care pur şi simplu copia ideile folosite de aceşti viruşi (dar şi de alții care au avut succese notabile). Ca rezultat apar anumite elemente comune care oferă un teren perfect de acțiune pentru un motor euristic bun. Când motoarele euristice şi abordarea generica sunt capabile să detecteze varietăți ale unui comportament dăunător cunoscut şi bine definit, companiile din domeniul AV nu trebuie să actualizeze fişiere de semnături odată cu fiecare versiune nouă de virus şi se pot concentra pe îmbunătățirea seturilor de reguli şi optimizarea produselor în aceeaşi linie.

În acelaşi timp se constată o continuă creştere a atacurilor din partea viruşilor cu un polimorfism din ce în ce mai avansat care de obicei nu pot fi detectați decât folosind o abordare algoritmica specifică motoarelor euristice. Având în vedere aceste aspecte este clar ca folosirea tehnicilor euristice în soluțiile antivirus este absolut necesară. Mai mult chiar, soluțiile antivirale nu sunt singurul domeniu în care putem utiliza tehnologii euristice. Este posibilă utilizarea unor astfel de abordări în detectarea încercărilor de intruziune în sisteme şi firewall-uri.

Bibliografie

1. Known Polymorphic Viruses, ftp.informatik.uni-hamburg.de, V. Bontchev, University of Hamburg 2002
2. Mit şi adevăr despre viruşii PC, J. Vasarhelyi, Z. Kasa, MicroInformatica Cluj – 1999
3. Computer Viruses – Theory and Experiments, F. Cohen, Elsevier Science Publishers B. V. - 1999
4. Security Focus Web Site, http://www.securityfocus.com/infocus, Marcus Schmall and colleagues