

ОӘЖ 004

**КОММЕРЦИЯЛЫҚ ПРОГРАММАЛЫҚ ҚАМТАМСЫЗ ЕТУДІҢ  
ҚАРАҚШЫЛЫғыНДАғы КЕРІ ИНЖЕНЕРИЯДАН ҚОРғанУ**

**Құрбан Ж.**

virtus30stm@gmail.com

Еуразия Ұлттық Университеті, Ақпараттық қауіпсіздіктің әдістері мен  
технологиялары мамандығының 2-курс магистранты, Нұр-Сұлтан қ.

Ғылыми жетекшісі – Ахметова Ж.Ж.

**Аннотация.** Қарақшылықтағы кері инженерия – бұл дайын программалық қамтамасыз етудің ішкі программалық кодын бақылау арқылы программаның жұмыс жасауын, кодтық архитектурасына қол жеткізуге мүмкіндік беретін процесстердің ортақ жиыны. Егер де қарақшыға программалық қамтамасыз етудің жұмыс жасау алгоритмдері белгілі болса, онда ол программаның лицензиялық кілт талап ететін қадамды заңсыз айналып өту мүмкіндігіне ие болады. Бұл ІТ-компаниялардың мансабы мен экономикасына үлкен нұсқан келтіретіндігі бір бөлек, тіпті мұндай іс-әрекеттерге заңды түрде қылмыстық іс қозғалады. Кері инженерияны жекелей талдайтын болсақ, бұл заңды іс-әрекет болып табылады. Процессты заңсыз ететін тек, ол субъектінің процессті қандай мақсатта жасап жатқанына байланысты ғана болады.

**Кілттік сөздер.** Программалық қамтамасыз ету; кері инженерия, обфускация, триггер, фреймворк, эксплойт, программалық код.

## **1. Кіріспе**

Осыдан 30-40 жыл бұрын программалық қамтамасыз етуді құрушылардың негізгі мақсаты – үлкен ғылыми орталықтардың тапсырыстарын орындау мақсатына бағытталған болса, қазіргі таңда программалық қамтамасыз етудің құрылуы және қолданылу аясы кеңейген. Қазіргі таңда программалық қамтамасыз етуді даярлайтын компаниялар табиғи ресурстарды өңдейтін компаниялармен экономикасы бойынша бір сатыда тұр. Есеп бойынша алынса, орта көлемдегі программалық қамтамасыз етуді құру үшін 5-6 адамнан тұратын команда жасақтауға 1 жыл шамасында уақыт кетеді екен. Барлық материалдық шығындар программалық қамтамасыз етудің сатылуымен тығыз байланысты болады. Егер программалық қамтамасыз ету жарық көрген соң оны тез арада заңды коммерциялық қорғауынан айырған болса, онда команда 1 жыл бойы бос жұмыс жасаған болмақ. Осындай программалаушылардың еңбегін көкке ұшырататын заңсыз процесс – программалық қамтамасыз етудің қарақшылығы деп аталады.

Бүгінгі күнде компьютерлік қарақшылық Қытай, АҚШ және Үндістанның үш ірі құқық бұзушылығының арқасында әлемдік проблемаға айналды. Қарақшылық программалық қамсыздандудың коммерциялық құны Солтүстік Америка мен Батыс Еуропада 19 миллиард долларды құраса, әлемнің басқа елдерінде бұл көрсеткіш 27,3 миллиард долларға дейін жетті. 2018 Global Software Survey зерттеуінде келтірілген мәліметтерге жүгінсек, дербес компьютерлерде орнатылған программалардың 37%-ы лицензияланбаған программалық қамтамасыз ету болып табылады [1]. Ал осы «қарақшылықты» іске асыруға қол жеткізетін құрал кері инженерия болып табылады.

## **2. Негізгі бөлім.**

Программалық қамтамасыз етудің қауіпсіздік саласынан қарастыруындағы ең негізгі фактор – ешқашан да қауіпсіздік шаралары 100 пайыздық қорғанысты бере алмайтындығында. Дегенмен компьютерлік қарақшылар үшін бұзу жұмыстарын қиындатуы және коммерциялық тұрғыда программалық қамтамасыз ету қажетті мөлшерде сатылып үлгеруі мүмкін.

Программалық қамтамасыз етудің қарақшылықтан қорғануын келесі кезеңдерге бөліп қарастыруға болады:

- Код обфускациясы.
- Тұтастықты тексеру.
- Программалық кодты шифрлау.
- Модульдік код архитектурасы.

### **Код обфускациясы**

Код обфускациясы – кодты оның функционалдығын сақтай отырып, талдауын қиындату процесі. Обфускация кері инженерия процесін едәуір күрделендіреді, себебі

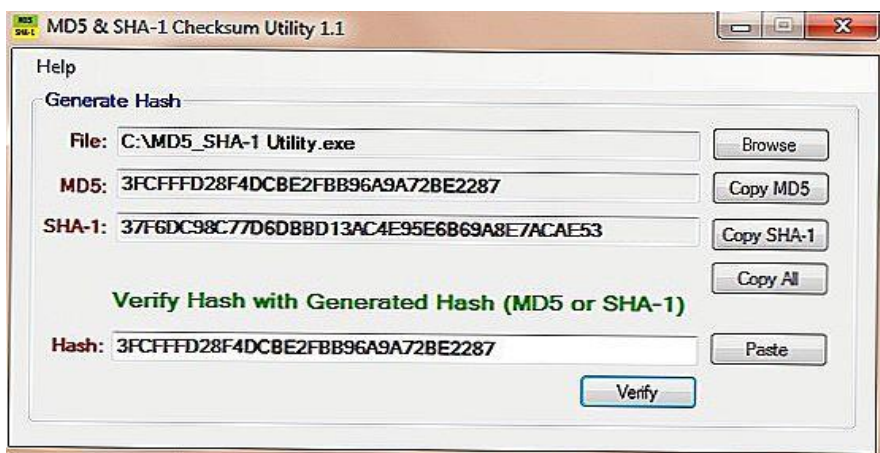
шабуылдаушыға бастапқы кодты алған жағдайда оның не істейтінін анықтау өте қиынға соғады. Обфускацияның ең тиімді түрлерінің бірі - мутация. Бұл программа жұмыс уақытында өзінің бастапқы кодын үнемі өзгертіп отырады, бұл кері инженерлік тапсырманы біршама ауырлатады. Алайда, мұнда проблемалар да кездеседі. Обфусцирленген код шабуылдаушы үшін ғана емес, программисттің өзі үшін де "оқылмайды"[2]. Сондай-ақ, кодтың қосымша тармақтарын қосу өнімділікті төмендетіп, тіпті қате кодына қосуына алып келуі мүмкін. Ең үлкен кемшілігі - шабуылдаушы бастапқы кодты алған жағдайда, тіпті қабылдау қиын болса да, обфускация жоғары қауіпсіздікке кепілдік бере алмайды. Шынында да, бұл жағдайда кодтың белгілі бір бөлімі мақсатты болып табылады, яғни лицензияны көшіруден немесе тексеруден қорғауды алып тастау үшін бүкіл қосымшаның жұмысын бөлшектеу қажет емес.

<p><b>До защиты:</b></p> <pre> .method private hidepinv instance void ExecuteOnDco() cil managed {   // Code size 41 (0x25)   .maxstack 9   locals init (class Calculator.CalcStack V_0, string V_1)   00: ldc.i4.0   01: ldc.i4.1 class Calculator.CalcStack Calculator.Calc::stack   02: stloc.0   03: ldc.i4.0   04: ldc.i4.1 class Calculator.CalcStack Calculator.Calc::operandStack   05: stloc.1   06: ldc.i4.0 class Calculator.CalcStack Calculator.Calc::stack   07: ldc.i4.0   08: ldc.i4.1 class Calculator.CalcStack Calculator.Calc::operandStack   09: stloc.2   10: ldc.i4.0 class Calculator.CalcStack Calculator.Calc::stack   11: stloc.3   12: ldc.i4.0 class [System.Windows.Forms]System.Windows.Forms.TextBox   13: ldc.i4.1 class Calculator.Calc::inputTextBox   14: callvirt instance string [System.Windows.Forms]System.Windows.Forms.TextBox::getText()   15: stloc.4   16: ldc.i4.0 class [System.Windows.Forms]System.Windows.Forms.Control   17: ldc.i4.1 class Calculator.Calc::inputTextBox   18: callvirt instance void [System.Windows.Forms]System.Windows.Forms.Control::setText(string)   19: stloc.5   20: ldc.i4.0 string Calculator.Calc::updateOutputString   21: callvirt instance void [System.Windows.Forms]System.Windows.Forms.Control::setText(string)   22: stloc.6   23: ldc.i4.0   24: ldc.i4.1 string Calculator.Calc::updateOutputString   25: call   26: call instance void Calculator.Calc::updateStackTextBox()   27: ldc.i4.0   28: call instance void Calculator.Calc::updateMemoryMenu()   29: ret   30: end   31: end of method Calc::ExecuteOnDco         </pre>	<p><b>После защиты:</b></p> <pre> .method private hidepinv instance void "()" cil managed {   // Code size 41 (0x25)   .maxstack 9   locals init (class "/I" V_0, string V_1)   21_0000: ldc.i4.0 class "/I/" "":   21_0001: stloc.0   21_0002: ldc.i4.0   21_0003: ldc.i4.1 class "/I/" "":   21_0004: stloc.1   21_0005: ldc.i4.0 class "/I/" "":   21_0006: stloc.2   21_0007: ldc.i4.0 class "/I/" "":   21_0008: stloc.3   21_0009: ldc.i4.1 class [System.Windows.Forms]System.Windows.Forms.TextBox "":   21_000a: callvirt instance string [System.Windows.Forms]System.Windows.Forms.TextBox::getText()   21_000b: stloc.4   21_000c: ldc.i4.0 class [System.Windows.Forms]System.Windows.Forms.Control   21_000d: ldc.i4.1 class [System.Windows.Forms]System.Windows.Forms.TextBox "":   21_000e: callvirt instance void [System.Windows.Forms]System.Windows.Forms.Control::setText(string)   21_000f: stloc.5   21_0010: ldc.i4.0 string "":   21_0011: callvirt instance void [System.Windows.Forms]System.Windows.Forms.Control::setText(string)   21_0012: stloc.6   21_0013: ldc.i4.0   21_0014: ldc.i4.1 string "":   21_0015: call   21_0016: call instance void "":   21_0017: ldc.i4.0   21_0018: call instance void "":   21_0019: ret   21_001a: end   21_001b: end of method "":         </pre>
--	---

1-сурет. Код обфускациясы

### Тұтастықты тексеру

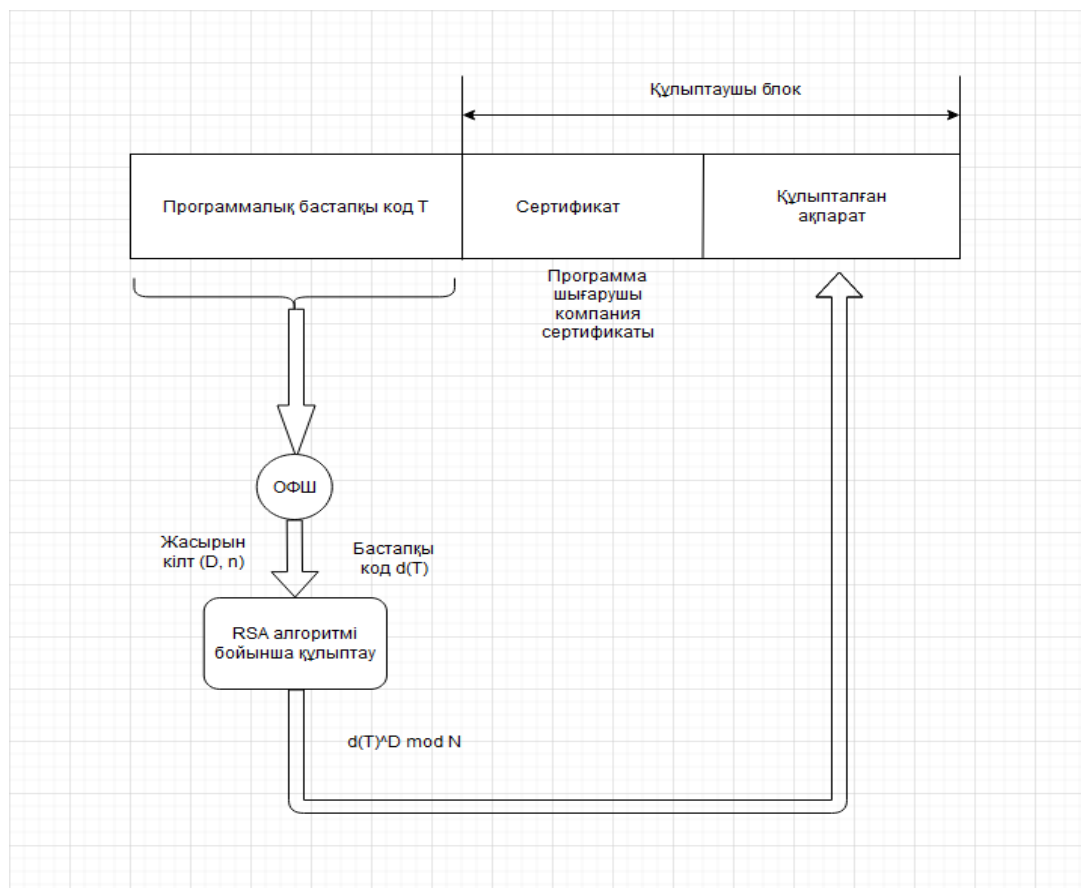
Тұтастықты тексеру – кодтың өзгертілмегенін растау процесі. Ол үшін программа кодының әр түрлі бөлімдерінің бақылау сомалары есептеледі және берілген мәнге сәйкес келмеген жағдайда қосымша жұмысын тоқтатады. Алайда, мұнда да қиындықтар туындауы мүмкін: егер шабуылдаушы қосымшаның бастапқы кодына қол жеткізсе, онда ол тұтастықты тексеруді алып тастай алады немесе оны әрқашан қажетті нәтижені қайтаратын функциямен алмастыра алады. Дегенмен тұтастық бақылау сомалары бұзуға оңай келмейтін хэш алгоритмдерімен жасалынатын болса (SHA-256), онда сәйкестендіру үшін де файлдардың саны оңайлыққа шыға қоймайды [3]. Тұтастыққа тексеру программалық қамтасыз етуді бірінші уақытта жүктеу кезінде де пайда береді. Бұл уақытта программалық қамтамасыз етудің заңды өкілеттіліктен алынғанына көз жеткізуге мүмкіндік береді.



2-сурет. Файл бақылау сомасын тексеру

### Программалық кодты шифрлау

Программалық кодты шифрлау – тек "заңды" сатып алушылар қосымшаны қолдана алатындығын тексеру, яғни шифрлау кілтінсіз программа жұмыс істемейді немесе тек сынақ тармақтарында жұмыс істейді. Алайда, мұнда кодтың қауіпсіздігіне ештеңе кепілдік бере алмайды, себебі кілттерді құру механизмін ашу мүмкіндігі бар [4]. Қазірде жаңашыл компиляторлар прекомпиляция кезінде шифр кілттерін енгізуге мүмкіндік береді. Кодты шифрлау қазірде ең маңызды қауіпсіздік шаралары болып табылады, себебі программалық қамтамасыз етудің бұзылуы тікелей шифр қиындығымен байланысты болады [5].



3-сурет. Программа кодын шифрлау

### Модульдік код архитектурасы

Модульдік код архитектурасы – бұл программалық қамтамасыз етуді жасау барысындағы қойылатын қауіпсіздік паттерні. Бұл паттерн бойынша коммерциялық программалық қамтамасыз етудің аса маңызды бөліктерін (лицензиялық кілт тексеру функционалы, «про» функционал [6], қауіпсіздік фреймворкі) жеке модульге компиляциялап, негізгі пакеттен бөлек жүктелетін болуы керек. Егер де негізгі пакет кері инженерия әдісі бойынша бұзылатын болса да, ол ешқандай пайда алып келмейді, оған себеп - заңды пайда алып келетін функционал ол уақытта жүктелмеген болуы мүмкін. Көптеген программалық қамтамасыз етудің лицензиялық кілт сұрау триггерін код отладкасы барысында аттап өтетін мүмкіндік туса, онда қарақшыға толыққанды программалық қамтамасыз ету тегін қолжетімді болады [7]. Аталынған паттерн осындай қауіпті жағдайларды өңдеуге мүмкіндік береді.

### 3. Қорытынды

Қауіпсіздік шараларын қорытындылай келе қорғаудың басқа әдістері де бар екендігіне көз жеткізуге болады, оларға су белгілері [8], кодтың маңызды бөліктерін жеке

модульдерге, қорғалған орындау орталарына шығару және т.б. жатады, бірақ олардың ешқайсысы толық қауіпсіздікті қамтамасыз ете алмайды. Қосымшаны қорғау деген көзқарас әр нақты жағдай үшін ерекше болуы керек. Мысалы, кодты өшіру қорғаныс құралы ретінде ғана емес, бұл әрекет кейбір жағдайларда өнімділікті арттыруы да мүмкін. Сонымен қатар, кодты бір жолға жазу немесе айнымалы атауларды неғұрлым қысқа және анық емес етіп ауыстыру программа жазу көлемінің төмендеуіне және қосымшаның өнімділігінің артуына әкеледі, алайда, код тармақтарын немесе бүркеншік аттарды қосу сияқты обфускацияның түрлері жұмыс жылдамдығын төмендетуі де мүмкін. Сондықтан кодты қорғаудың әдістерін таңдағанда, ең алдымен, қауіп-қатер моделін басшылыққа алу керек, атап айтқанда: қосымшада не қорғау керек және шабуылдаушы оны қалай алуға тырысады деген сұрақтарға жауап іздеу қажет. Егер бұл кодты өзгерту болса, онда тұтастықты тексеруге назар аударылу қажет, ал егер қосымшаның бір бөлігі зерттелінсе, обфускация немесе шифрлау опциясын қарастырған жөн.

#### **Қолданылған әдебиеттер**

1. Panda Security, What is Software Piracy?, 2019, <https://www.pandasecurity.com/mediacenter/panda-security/software-piracy/>
2. Zander A; Digital and Software Piracy, 2015, 11-14 б.
3. Zander A; Digital and Software Piracy, 2015, 97-101 б.
4. Craig P., Software Piracy Exposed, 2005, 73-74 б.
5. Dowd M; The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities, 503-510 б.
6. Панов А; Реверсинг и защита программ от взлома, 2006, 20-21 б.
7. Панов А; Реверсинг и защита программ от взлома, 2006, 55-27 б.
8. Хогланд Г; Взлом программного обеспечения, 2005, 310 б.