Аталған құралдардың жұмысының мәні белгілі бір ережелер жиынтығына сәйкес болу принципі бойынша блоктауда жатыр. Бұл ретте трафикті талдау жүргізілмейді. Басқаша айтқанда, бұл бағдарламалар бөлінген шабуылдарға қарсы іс-қимыл үшін тиімсіз.

**Қорытынды:** Қызмет көрсетуден бас тартуға бағытталған таратылған желілік шабуылдардың принциптері зерттелді. Сондай-ақ соңғы уақытта жүргізілетін кейбір DDoS-шабуылдарға мониторинг жүргізілді. Эксперимент нәтижелері кестеде көрсетілген.

| Жүйе | Жалған іске қосу, % | Анықталмаған зиянды сұраулар, % | Шабуылдың басталуы мен табылуы арасындағы орташа уақыт |
|---|---|---|---|
| Kaspersky Anti-Haker | 0,7 | 11 | 44 минут |
| Snort | 3,8 | 10,9 | 17 минут |
| Symantec | 4,3 | 8,4 | 12 минут |

**Қолданылған әдебиеттер тізімі**


1. DDOS атаки [Электронный ресурс]. — URL: http:// localname.ru/soft/ataki-tipa-otkaz-v-obsluzhivanii-dos-iraspredelennyiy-otkaz-v-obsluzhivanii-ddos.html.
2. Предотвращение атак с распределенным отказом в обслуживании (DDoS) [Электронный ресурс] / Официальный сайт компании Cisco. — URL: http:// www.cisco.com/web/RU/products/ps5887/products_white_ paper0900aecd8011e927_.html.
3. Методы защиты от DDOS нападений [Электронный ресурс]. — URL: http://www.securitylab.ru/analytics/216251.php.
4. Терновой О.С., Шатохин А.С. Снижение ошибки обнаружения DDOS атак статистическими методами при учете сезонности // Ползуновский вестник. — 2012. — №3/2.
5. Бенкен Е.С. PHP, MySQL, XML. Программирование для Интернета. — СПб., 2011.


ӘОЖ 004.454

## ТҮРЛІ ОПЕРАЦИЯЛЫҚ ЖҮЙЕЛЕР ДРАЙВЕРЛЕРІН ВЕРИФИКАЦИЯЛАУ ЖҮЙЕЛЕРІ


**Мухамедиева Лаура Сеилхановна**
*l-muhamedieva@mail.ru*
Л.Н.Гумилев атындағы ЕҰУ Информатика және ақпараттық жүйе магистранты, Астана, Қазақстан
Ғылыми жетекшісі – т.ғ.к., доц. Р.Д. Туребаева


Операциялық жүйе (ОЖ) драйверлерін верификациялаудың барлық жүйелері ОЖ ядросымен драйверлердің өзара әрекеттесу ережелерін бұзумен байланысты қателерді іздеуге арналған. Осыған байланысты, драйвердің арнайы ортасын дайындау қажет, бұл драйверді өңдеушінің шақыру тәртібіне қойылатын шектеулерді ескеруге мүмкіндік береді. Верификациялау үшін бағдарламаның еркін нүктесінің жетістіктерін тексеруге мүмкіндік беретін статикалық верификация құралдары қолданылады. Әр түрлі ОЖ ядросының модульдерін верификациялау процесін автоматтандыру үшін статикалық верификация жүйесі әзірленеді.

Операциялық жүйелердің драйверлерін верификациялаудың ең дамыған жүйесі *Microsoft Static Driver Verifier (SDV) жүйесі* болып табылады [1], ол Microsoft Windows операциялық жүйесінің драйверлерін статикалық верификациялауды жүргізуге асыруға мүмкіндік береді. Ол драйверлерді сертификаттау процесінде қолданылады және 2006 жылдан бастап Microsoft Windows Driver Developer Kit құрамына енгізілді. Microsoft SDV жүйесі нақты бағдарламаларды верификациялауды қолдану мүмкіндігін көрнекі көрсетеді.

SDV жүйесінде пайдаланушыдан қоршаған ортаны құру үшін, өңдеушілердің әрбір функциясының рөлін көрсете отырып, драйверлердің бастапқы кодын қолмен аннотациялауды талап етеді.

Аннотацияларды қолмен жасау керек, алайда, SDV қолданудан оң нәтиже аннотациялауға еңбек шығындарын өтейді, сондықтан көбінесе аннотацияларды драйверлерді әзірлеушілер тікелей жазады. SDV тәсілі шеңберінде қоршаған ортаның моделі ретінде өңдеушілер функциясының түрлеріне сәйкес шақырулардың еркін реттілігі жасалады. Генерацияланатын тізбекте өңдеушілердің түрлеріне сәйкес функцияларды шақыру тәртібіне қойылатын шектеулер ескеріледі. Microsoft Windows ОЖ ядросының негізгі бөлігінің функциялары қарапайым аналогтармен модельделеді, және де олар үшін әдетте детерминацияланбаған модельдер қолданылады. Сонымен қатар, статикалық талдау құралы недетерминизмнің барлық тармақтарын зерттейді, сонымен сирек кездесетін функцияларды қоса алғанда, функциялардың барлық әр түрлі нұсқаларын тексереді.

Тексерілетін әдептілік ережелері SLIC тіл көмегімен қалыптастырылады, онда драйвердің бастапқы кодымен байланыс ядро функциясының шақыруларын ұстап тұратын аспект-бағытталған конструкциялардың көмегімен орнатылады. Бұл тәсілде қате моделін әзірлеу үшін Си бағдарламалау тіліне өте ұқсас қарапайым арнайы тіл беріледі. Бұл тіл функциялардың мінез-құлқын моделдеуге ғана бағытталған, сондай-ақ бағдарламаның жаһандық жағдайын сақтауға мүмкіндігі бар. Жалпы айтқанда, осының салдарынан қателерді модельдеу үшін қажетті мәнерлі қуат өте қатты зардап шегеді.

```
state { int zero_cnt = 0; }
put.entry {
  if ($1 == 0) {
    if (zero_cnt == 4)
      abort "Queue has 4 zeroes!";
    else
      zero_cnt = zero_cnt + 1;
  }
}
get.exit {
  if ($return == 0)
    zero_cnt = zero_cnt - 1;
}
```

Сурет 1. SLIC үлгі ерекшелігі

Алайда, тілдің қарапайымдылығы оны верификация саласындағы сарапшылар емес пайдаланушыларға да қате модельдерін құру үшін пайдалануға мүмкіндік береді. Қазіргі уақытта шамамен 200 ережелердің жинағы таңдалған, ал зерттеу нұсқасында жаңа ережелерді қосу мүмкіндігі қосылды.

Келтірілген 1 суретте көрсетілген SLIC тіліндегі ерекшелігі жасанды қатені тексеруді көрсетеді.

Бұл жағдайда, кезекте төрт нөлден артық жағдай қате болып табылады. SLIC препроцессоры спецификация негізінде драйверлердің бастапқы кодын құралдайды.Нәтижесінде, құрал Си-ге баламалы бағдарламаны генерациялайды, содан кейін кодты статикалық талдау құралы арқылы тексеріледі.

Өкінішке орай, SLIC тек Microsoft Windows ОЖ драйверлері үшін спецификация мен құрал-саймандарды жасау үшін пайдалануға болады. Сонымен қатар, SDV жүйесін іске асырудың бастапқы коды толығымен жабық. Жүйеге SLAM және Yogi статикалық верификация құралдары біріктірілген.

Сонымен қатар, Windows-тің драйвері мен ОЖ арасындағы интерфейс өте сирек өзгереді, жүйенің архитектурасын әзірлеу кезінде драйверлер мен ОЖ ядросы арасындағы интерфейстің тұрақты даму мүмкіндігін ескеруді талап етпейді.

Қазіргі уақытта Linux ОЖ үшін SDV өлшемімен салыстырылатын верификация бойынша индустриялық жоба жоқ. Бірақ әзірлеушілер қауымдастығы осы ОЖ жұмысының ең күрделі аспектілерін автоматты түрде верификациялау құралдарын үнемі іздейді, ал верификация саласындағы зерттеушілер қоғамдастығы жаңа талдау тәсілдерін қолдану үшін Linux драйверлерін пайдаланады. Соңғы жылдары ОС Linux драйверлері негізінде үш верификация ортасы құрылды: Linux Driver Verification, Avinux және DDVerify [2].

*Avinux жүйесі* Тюбинген қаласының (Германия) университетінде жасалған. Avinux ядро модульдерінің бірлі-жарым препроцессирленген файлдарын автоматты түрде тексеруге мүмкіндік береді, ол үшін онда ядро жинау процесіне жинақтауды сипаттайтын түпнұсқалық скрипттерді түрлендіру жолымен кірістіру жүзеге асырылады. Avinux ортасының моделі іс жүзінде толық қолмен қойылады (параметрлері үшін инициализация коды жасалатын модульдердің экспортталатын функциялары ғана автоматты түрде шақырылады). ОЖ ядросының бағдарламалық интерфейсін дұрыс пайдалану ережелерінің спецификациялары SLIC тілін кеңейтуде жазылады. Avinux- де синхрондау примитивтерін дұрыс пайдалану ережелерін және жадымен дұрыс жұмыс істеу ережелерін тексеруді қолдайды. Статикалық верификация жүйесі CBMC статикалық верификациясының жалғыз құралымен біріктірілген. Avinux Eclipse үшін плагин түрінде іске асырылған, бұл жүйені автоматтандырылған түрде іске қосуға мүмкіндік береді. Бірақ пайдаланушыға CBMC форматындағы қателердің трассалары беріледі, бұл олардың талдауын қиындатады.

*DDVerify жүйесі* Оксфорд университетінде (Англия) жасалған. DDVerify талданатын модульдерді құрастырудың құрамы мен опциялары туралы ақпаратты алу үшін жеке жинау скрипттерін пайдаланады. Статикалық верификацияның осы жүйесіндегі келісімшарттық ерекшеліктер Си бағдарламалау тілінде толық қолмен беріледі . DDVerify әзірлеушілері модульдердің төрт түріне, сондай-ақ аппараттық үзіктерді , таймерлерді және т. б. өңдеушілерге арналған қоршаған ортаны жасады. Сонымен қатар, олар синхрондау примитивтерін дұрыс пайдалану ережелерінің және оларды пайдаланғанға дейін айнымалыларды дұрыс инициалдау ережелерінің сегіз спецификациясын қойды. DDVerify модулдерді статикалық верификациясының CBMC және SATABS екі құралы арқылы тексеруге мүмкіндік береді. Қателердің трассаларын талдауды жеңілдету үшін статикалық верификация жүйесінің құрамына Eclipse интеграцияланған әзірлеу ортасы үшін плагин кіреді.

*Linux Driver Verification LDV* жүйесі РАН Жүйелік бағдарламалау институтында (Ресей) жасалған. LDV жүйесі ядроны жинау процесімен біріктірілген, сондықтан драйверлердің құрамы мен құрастыру параметрлері туралы барлық қажетті ақпарат автоматты түрде алынады. Қоршаған ортаны генерациялау драйверлердің барлық түрлерін жабатын үлгілердің иерархиясы негізінде толығымен автоматты түрде жүзеге асырылады . Жүйе Си бағдарламалау тілін аспект-бағдарлы кеңейту арқылы түзетудің жаңа ережелерін қосуға мүмкіндік береді. Linux ОЖ құрылғылар драйверлерін верификациялау ескеру қажет бірқатар ерекшеліктерге ие. Негізгі ерекшеліктердің бірі - драйвер мен Linux ОЖ ядросы арасындағы интерфейстер үнемі өзгереді. Бұл ядро құжаттамасында белгіленген Linux ОЖ әзірлеушісінің принципиалды позициясы болып табылады.

Linux ОЖ құрылғылар драйверлерін верификациялау бірқатар ерекшеліктерге ие. Негізгі ерекшеліктердің бірі- драйвер мен Linux ОЖ ядросы арасындағы интерфейстер үнемі өзгереді. Бұл ядро құжаттамасында белгіленген Linux ОЖ әзірлеушісінің принципиалды позициясы болып табылады . Бұл, мысалы, жүйенің жұмысын жеделдету үшін, өзара іс-қимыл интерфейстерін жетілдіру мүмкіндіктерінде әзірлеушілер өздерін шектемейді. Осылайша, Linux ядросы үнемі дамиды, өзара әрекеттесу ережелері өзгереді, жаңа пайда болады, драйвер ортасы өзгереді. Сондықтан верификация жүйесінің ережелері, қоршаған ортаның үлгілері және басқа да бөліктері даму қарапайымдылығын, ОЖ ядросының ағымдағы жағдайына

бейімделуін, драйверлер мен ОЖ арасындағы өзара іс-қимыл ережелерін қамтамасыз ететіндей болуы тиіс. Бұл ерекшеліктер Linux ОЖ драйверлерін верификациялау үшін Microsoft SDV жүйесінің архитектурасын пайдалануға мүмкіндік бермейді.

Еркін бағдарламалық қамтамасыз ету әлеміндегі бағдарламалық құралдардың өмір сүру уақытын болжау қиын. Демек, жүйенің өзгерістерге ашық болуы, атап айтқанда, ескірген немесе қолдаудан айырылған верификацияның жаңа құралдарын біріктіруге дайын болуы маңызды.

Сондықтан, Linux ОЖ драйверлерін верификациялау ерекшеліктерін ескеретін статикалық верификация жүйесінің әдісі мен архитектурасы драйверлерді верификациялаудың ең тиімді және ығайлы әдісі болып табылады.

**Қолданылған әдебиеттер тізімі**

1. Ball T., Bounimova E., Levin V. et al. The Static Driver Verifier Research Platform // Computer Aided Verification. CAV'10. 2010. P. 119–122.
2. Мандрыкин М. У., Мутилин В. С., Новиков Е. М. и др. Использование драйверов устройств операционной системы Linux для сравнения инструментов статической верификации // Программирование. 2012. Т. 5. С. 54–71.

ОӘЖ 004

## ӘДІСТЕМЕ ҰҒЫМЫНЫҢ МӘНІ ЖӘНЕ ЖОБАЛАРДЫ БАСҚАРУДЫҢ ӘДІСТЕМЕСІ

**Мырзахан Нұрбибі Нұрболатқызы**
*Nurbi0195@gmail.com*
Л.Н.Гумилев атындағы ЕҰУ Ақпараттық технологиялар факультеті, Ақпараттық жүйелер мамандығының 2-курс магистранты, Нұр-Сұлтан, Қазақстан

**Аннотация.** Қазіргі қоғамда көбінесе тек бизнесте ғана емес, қарапайым өмірде де адамның жұмысының тиімділігі туралы сұрақ туындайды. Табысты жобалардың пайызы біз қалағандай көп емес. Бұл мақалада қойылған мақсатқа жетудің көптеген әдістері жасалып енгізілді. Алайда, оларды жеке жобаны сәтті іске асырудың негізгі принциптері мен әдістері, оны жүзеге асыруда нақты болатындай талдауға және нақтылауға болатын бірыңғай әдістемемен біріктірілмеген.

**Кілттік сөздер:** әдістеме, жобаларды басқару, жоба, жеке жоба.

Ғылымды дамытудың әртүрлі кезеңдерінде ғылыми зерттеулерді әдістемелік негіздеу мәселелері әрдайым маңызды орын алды. Кез келген ғылымды дамыту оны жаңа фактілермен толықтыру арқылы жүзеге асырылады, оларды түсіндіру зерттеудің ғылыми негізделген әдістерін қолдану арқылы қамтамасыз етіледі.

Әдістеме нақтылықтың кез-келген құбылысын танудың құралы ретінде әрекет етеді және оның негізгі бағыттарын анықтайды.

Қазіргі гуманитарлық ғылымдардағы әдістеменің түсінігі әртүрлі аспектілер бойынша түсіндіріледі, олар төменде талқыланады. Әдістеменің анықтамасынан бастайық («әдіс» және «логика») - «әдіс туралы ілім», «әдіс теориясы». Кең мағынада оны барлық пәндерге ортақ ғылыми білімнің бастапқы ұстанымы ретінде түсіндіруге болады. Тар мағынада әдістемені белгілі бір пән бойынша ғылыми таным теориясы ретінде қарастыруға болады [2]. Философиялық энциклопедиялық сөздікке сәйкес әдістеме дегеніміз - теориялық және практикалық іс-әрекетті ұйымдастыру мен құрудың принциптері мен әдістерінің жүйесі, сонымен қатар осы жүйенің ілімі [1]. Әдістеменің негізгі міндеттеріне мыналар кіреді: негізгі зерттеу бағдарламалары мен бағыттарын айқындау, мәселелердің дұрыс тұжырымдалуын қамтамасыз ету, ғылыми зерттеулердің құралдары мен әдістерін әзірлеу, тақырыпты зерттеудің жалпы тәсілдерін табу, тақырыптың мазмұнына сәйкес зерттеу кезеңдерін реттеу,