УДК 004.056.53.001.3

Соколов Артем Михайлович

Сучасні моделі виявлення аномалій в комп’ютерних системах

Вступ

Останнім часом світ переконався, що навіть найнадійніші системи захисту не здатні захистити від атак комп’ютерні системи державних і комерційних установ. Одна з причин — у тому, що в більшості систем безпеки застосовують стандартні механізми захисту: ідентифікацію та аутентифікацію, механізми обмеження доступу до інформації згідно з правами суб’єкта і криптографічні механізми. Це традиційний підхід із своїми недоліками, як-то: незахищеність від власних користувачів — зловмисників, розмитість поділу суб’єктів системи на "своїх" і "чужих" через глобалізацію інформаційних ресурсів, порівняна легкість підбору паролів внаслідок використання їхнього змістового різновиду, зниження продуктивності і ускладнення інформаційних комунікацій внаслідок обмеження доступу до ресурсів організації. Звідси потреба в механізмах, які б, доповнюючи традиційні, уможливлювали виявлення спроб несанкціонованого доступу і інформували про це відповідальних за безпеку або реагували у відповідь. Важливо, щоб такі системи могли протистояти атакам, навіть якщо зловмисник уже був аутентифікований та авторизований і з формальної точки зору додержання прав доступу мав необхідні повноваження на свої дії.

Ці функції і виконують системи виявлення вторгнень (*intrusion detection systems*, *IDS*). Оскільки передбачити всі сценарії розгортання подій в системі з активним "чужим" суб’єктом неможливо, слід або якомога детальніше описати можливі "зловмисні" сценарії або ж, навпаки, —"нормальні" і постулювати, що всяка активність, яка не підпадає під прийняте розуміння "нормальності", є небезпечною. Згідно з цим, IDS поділяються на системи, що реагують на відомі атаки — системи виявлення зловживань (*misuse detection systems, MDS*) і системи виявлення аномалій (*anomaly detection systems, ADS*), які реєструють відхилення еволюції системи від нормального перебігу.

Системи виявлення зловживань. Атака є багатокроковим процесом, і його здійснення потребує високої кваліфікації хакера. Тому найпростішим шляхом злому або приведення системи в недієздатний стан є застосування "експлойтів", тобто вже написаних модулів, що реалізують необхідні етапи атаки. Величезна кількість експлойтів доступна через Internet, що призвело до поширення саме такого способу атаки. Як наслідок, часто послідовність аудит-подій, що реалізують атаку є фіксованою.

Робота MDS базується на складанні шаблонів таких атак. Їхній рівень абстракції може бути простим, як наявність певних значень у заголовку мережевого пакету або послідовності команд у файлі аудиту, так і складним, як проходження траєкторії системи в просторі станів через певні небезпечні стани. Захисні системи такого типу ефективні на відомих схемах атак, проте у випадках невідомої атаки або відхилень перебігу атаки від шаблону виникають проблеми, а тому слід підтримувати велику базу даних на кожну атаку та її варіації і організовувати безперервне поповнення баз шаблонів.

Системи виявлення аномалій. Основним припущенням ADS є те, що дії зловмисника (події в атакованій системі) обов'язково чимсь відрізняються від поведінки звичайного користувача (від подій в нормальному стані), тобто є аномаліями. Тому такі системи здатні реєструвати і невідомі атаки. Роботі ADS передує період накопичення інформації, коли складається концепція нормальної активності системи, процесу чи користувача. Вона стає еталоном, відносно якого оцінюються наступні дані. Тут визначається оптимальна кількість факторів, за якими вестимуться спостереження. Їх сукупність не повинна бути надто великою, бо це знизить продуктивність роботи в цілому. Вона також не повинна бути надто обмеженою, оскільки за недостатньо вичерпними характеристиками неможливо буде побудувати профіль нормальної поведінки.

Можливі два загальні види помилок: а) нормальна поведінка системи або користувача помилково приймається за зловмисну (*false positives*); б) спроба зловмисного проникнення в систему приймається за нормальну активність (*false negatives*). Хоча жодна з цих ситуацій

небажана, друга — більш небезпечна, і тому однією з основних задач побудови ADS є чітке визначення умов, за яких ситуація сприймається як аномальна, так, щоб жодна з перелічених ситуацій не виникала занадто часто [1].

Моделі виявлення аномалій

Піонерською роботою в цій галузі була робота [2], в якій наводилися основні поняття і підходи до проблеми. Пізніше в літературі з'явилося чимало спроб побудови ADS. Більшість із них — концептуальні моделі, мета яких — перевірити можливість застосування математичної моделі чи підходу. Комерційних продуктів на царині IDS дуже мало, а ті, що є, майже ніколи не виходять за рамки MDS.

Практично всі описані в літературі моделі для виявлення аномалій можна розділити на: а) ті, що базуються на зберіганні прикладів поведінки; б) частотні; в) нейромережеві; г) ті, що будують скінченні автомати; д) інші спеціальні.

Методи, що базуються на зберіганні прикладів поведінки. Найпростішим підходом є пряме запам'ятовування прикладів дій, послідовностей команд користувачів чи взагалі будь-яких параметрів доступних реєстрації (*instance-based learning*). Попри неможливість застосування цього підходу в інших випадках моделювання людської поведінки, у задачах виявлення вторгнень він є достатньо дієвим. Що зумовлено обмеженою кількістю можливих дій суб'єктів комп'ютерної системи, значною детермінованістю задач, що можна виконати і самою структурою операційної системи.

У роботі [3] проводиться аналогія з гомеостазом — процесом в живих організмах, що підтримує їх у життєдіяльному стані. Реакцією "організму" (комп'ютерної системи) на аномальну поведінку процесу є його примусове уповільнення. Таким чином, процеси, що демонструють жваву аномальну поведінку, будуть майже зупинені і автоматично знищені системою як такі, що не реагують на запити. Зафіксовані раніше підпослідовності системних визовів, що зустрічалися в тренувальній множин, запам'ятовуються і під час роботи перевіряється їх наявність в поточній сесії. Як і в [4], стверджується, що оскільки спостереження ведеться за системними визовами

програми, то через високу їх регулярність (послідовність визовів і їх типи значно детерміновані вихідним кодом програми) розміри бази підпослідовностей не будуть великими. Підпослідовність з поточної сесії, якої не було серед тренувальних, вважається аномальною. Підхід потребує дописування ядра операційної системи, що є нелегкою і не завжди можливою задачею. Крім того, постійна присутність такого моніторингового компонента призводить до загального уповільнення роботи всієї системи, яке складає 4% – 50% [3].

Іншим зразком instance-based системи є [5]. Вводиться спеціальна метрика схожості символьних послідовностей. Так, для послідовностей X та Y їхня схожість $Sim(X,Y) = \sum_{i=0}^{l-1} w(X,Y,i)$, де $w(X,Y,i)=1+w(X,Y,i-1)$, якщо $x_i=y_i$, і $w(X,Y,i)=0$, якщо $x_i \neq y_i$ або $i<0$. Для кожного користувача зберігається множина $\boldsymbol{D}$ послідовностей, які він звичайно виконує. За міру схожості множини $\boldsymbol{D}$ і послідовності, що надійшла для перевірки, приймається величина $Sim_D(X) = \max_{Y \in D}\{Sim(X,Y)\}$. Через необхідність збереження великої кількості даних на кожного користувача виникає потреба у застосуванні спеціальних методів зменшення об’єму баз послідовностей. Автори розглядають два таких методи, а саме: вибіркової селекції прикладів, коли зберігаються не всі приклади послідовностей, а тільки останні n, або всі, крім n з найменшими ймовірностями, та перетворення бази на базу представників класів елементів.

Через високі вимоги до ресурсів instance-based системи можуть застосовуватися лише в задачах виявлення аномалій з незначною кількістю можливих сигналів та в основному із статичною поведінкою суб’єктів спостереження.

Частотні моделі. Розвиненням ідей instance-based систем є прийняття до уваги частотного розподілу параметрів системи. Вже в [2] пропонувалося зберігати інформацію про суб'єктів у шаблонах активності — виражених у статистичних термінах наборах характеристик поведінки суб'єкта відносно певного об'єкта, як-то: входження до системи, запуски програм, доступи до файлів і пристроїв, з метою реєстрації відхилень. Потім перевіряється чи попадає відносна кількість певних подій в заданий експертом інтервал.

Модифікацією частотного підходу є робота [6], де пропонується метод, що базується на т.зв. структурних нулях. Він полягає у використанні інформації про команди, які використовують дуже рідко або зовсім не використовують (відповідні їм комірки в таблиці ймовірностей дорівнюють нулю, тобто є структурними нулями). Вводиться індекс унікальності $\frac{1}{N_i}\sum_{c\in\Sigma} W_{ic}U_cN_{ic}$, що обчислюється для кожної сесії та кожного користувача *i*. Тут N_i — кількість команд у сесії, W_{ic} — індикатор {1,-1} присутності команди *c* в тренувальних даних користувача, N_{ic} — кількість входжень команди *c* в поточну сесію і U_c — параметр, що змінюється від 0 для команди, що використовувалася всіма користувачами, до 1 для команди, яка не була використана жодним користувачем. Цей індекс отримує додатний приріст для частих команд у межах поточної сесії, який тим менший, чим частіше ця команда використовується взагалі. Так, широке використання загалом рідких команд спричинить великі значення індексу унікальності. У разі появи команд, не властивих користувачеві, індекс зменшується. Припустивши, що значення індексу є стабільним для даного суб'єкта, автори намагаються розрізняти їх за значеннями індексу.

Поширеними недоліками частотних методів є неадаптивність, оскільки часто еталонні значення частот визначаються одноразово, за тренувальною множиною або за експертними даними, і невраховування послідовності виконання команд.

Нейромережеві моделі. Застосування нейронних мереж зумовлюється самою неформальної постановкою задачі — виявити аномальну поведінку. Ідея полягає в тому, щоб, отримавши деяку "тренувальну" множину параметрів вхід-вихід, що характеризують поведінку системи, дати мережі "звикнути" до них. Виходом може бути деякий коефіцієнт "нормальності" поведінки або один із параметрів системи. Якщо вхідні дані мають закономірності, то робиться припущення, що мережа здатна "навчитися" на них. Якщо в процесі роботи запропонований нейронною мережею вихід, якщо він є деяким коефіцієнтом, попадає в небезпечну область або відрізняється від наявного в реальній системі за умови, що це один із параметрів системи, то робиться висновок, що в системі наявна аномалія.

В роботі [7] для побудови шаблону поведінки користувача використовуються такі параметри: години, коли він зазвичай працює, набір вузлів, з яких він починає робочу сесію, характеристики використання ресурсів системи тощо. Ці параметри оцифровуються і слугують входом до нейронної мережі зворотного поширення помилки (*backpropagation neural network*, *BPNN*), а виходом є коефіцієнт, що дорівнює нулю для користувача з нормальною поведінкою і рівний одиниці — з аномальною. Іншими словами, мережа тренується на парах типу ("нормальні" параметри, 0) та ("аномальні" параметри, 1). Оскільки для отримання "ненормальної" поведінки слід було б примусити користувача поводитися не так, як він звик, то аномальні дані генеруються випадково, що ускладнює інтерпретацію результатів відносно роботи з реальними даними.

У роботі [8] запропонована система ідентифікує користувачів на основі обмеженої кількості команд (всього 100). Враховується кількість запусків кожної команди, а не їхня послідовність. Ці певним способом закодовані кількості слугують за вхід до BPNN, на яких вона тренується з учителем. Вихід — ідентифікатор користувача (параметр, який, хоч і має в UNIX числове значення, але створює штучну близькість користувачів з близькими ідентифікаторами). Якщо користувача ідентифіковано неправильно — на це звертається увага адміністратора. Були отримані непогані результати, але лише з 10 користувачами. Такі умови є "тепличними", оскільки в реальних системах кількість користувачів досягає кількох тисяч, причому більшість виконує

однотипні дії, що ускладнює їх розрізнення таким способом.

У нашій попередній роботі [9] теж було запропоновано підхід на основі BPNN. На відміну від [7,8], ми не обмежувалися вузьким колом користувачів, кількість команд була більшою (до 512 команд) і не використовували штучних даних. Мережа застосовувалася для передбачення наступної (n+1)-ої команди в сесії користувача, використовуючи попередні n команд. Через невпорядкований характер множини команд було запропоновано компромісне кодування команд: кожній команді ставився у відповідність вектор — двійковий запис номера команди після довільного перенумерування всіх команд. Таким чином, ми частково позбувалися проблеми штучної впорядкованості, коли командам надавали скалярних значень і діставали досить економний метод кодування (на відміну від методу, коли окремій команді присвоюється вектор, який має одну одиницю у фіксованій позиції). Після тренування на входах, що відповідали контексту попередніх декількох команд, які потрапляли у вікно ширини n і з виходом — наступною командою, мережа повинна була на реальних даних за наявним контекстом визначати наступну. Якщо відносна кількість правильно передбачених команд у сесії не перевищувала заданий рівень, це сигналізувало про аномалію. Для забезпечення адаптивності нейронна мережа після кожної сесії дотреновувалася. Результати свідчать про можливість застосування такого підходу і про ефективність обраного методу кодування команд, проте нерегулярність поведінки користувачів значно підвищує рівень помилок типу false positives.

Очікується, що кращі результати можуть бути отримані на даних від регулярніших джерел, як-то системні процеси. Додаткові приклади застосування BPNN і рекурентних нейронних мереж для виявлення аномалій наведено в [10-12]. Недослідженими залишаються можливості застосування нейронних мереж неперсептронного типу, наприклад ансамблевих.

Недоліком багатьох нейронних мереж є їхня погана пристосованість для роботи з невпорядкованими величинами. Введення штучного порядку на множині значень елементів тільки спотворить картину, оскільки нейронна мережа "враховуватиме" близькість числових величин.

Моделі, які будують скінченні автомати. В цьому підході досягається більша моделююча здатність, ніж при використанні тривіальних частотних та instance-based методів. Вхідні дані розглядаються як потік дискретних подій, наприклад системних визовів або ідентифікаторів процесів. Мета — отримати автомат, який моделює вказану послідовність подій. Для багатьох послідовностей характерне те, що ймовірність наступного символу, елементу або сигналу залежить від попередніх. Часто вони залежать лише від невеликої кількості попередніх. Це наштовхує на думку моделювати їх за допомогою марківських ланцюгів [2,4,13-15]. Проте при зростанні порядку ланцюгів, яке може суттєво збільшити точність моделі, кількість станів відповідного автомата поводить себе як $O(\Sigma^L)$, де Σ — розмір алфавіту символів, L — порядок ланцюга. Це висуває великі вимоги до ресурсів і збільшує час обробки.

Роботи [13,15] є прикладами найпростішого застосування марківських ланцюгів до моделювання процесів у комп'ютерній системі. За вхідними даними будується матриця переходів ланцюга першого порядку і ймовірність сесії визначається як добуток імовірностей переходу між станами, що відповідають елементарним подіям у файлі аудиту.

Більш потужною моделлю є т.зв. приховані марківські моделі (HMM). Від марківських ланцюгів вони відрізняються тим, що вихідні символи автомата не детерміновані його станами, а залежать від них стохастично. Є теоретичні результати, які свідчать про складність навчання HMM. Крім того, часто необхідний підбір кількості прихованих станів, який звичайно проводиться вручну і після тренування реально задіяними (такими, що мають відмінні від нуля ймовірності переходів) виявляються лише деякі з них. Підхід до виявлення аномалій на основі HMM пропонується, наприклад, в роботі [16]. Для забезпечення адаптивності необхідно періодично перенастроювати HMM на нових даних, що є значним недоліком моделі.

В роботі [17] скінченний автомат будується за *n*-грамами подій з аудит-файлу, який не містить аномальних даних. У першому варіанті побудови кожному стану відповідає одна або кілька *n*-грам. Кілька *n*-грам можуть бути присвоєні одному стану, якщо при послідовній обробці даних, вікно за вікном, маємо ситуацію, коли наступна *n*-грама ще не має свого стану і одночасно

не існує ребра, що виходить з поточної *n*-грами в наступну. Тоді наступна *n*-грама присвоюється тому ж станові, що й поточна. Побудований у такий спосіб недермінований автомат може знаходити раніше не бачені події, проте він не бере до уваги статистичні параметри послідовності. За другим способом будується імовірнісний автомат з множиною входів — подіями з аудит-потоку і множиною виходів — послідовністю чисел, що вказують на ступінь аномальності вхідних подій. Ці числа показують, як імовірність наявного переходу з поточного стану в наступний співвідноситься з вектором перехідних імовірностей з цього стану. Попри деяку необґрунтованість вибору саме такої міри аномальності підхід на експерименті виявився ефективним.

Удосконалюючи модель для виявлення аномалій за системними визовами (згадаємо фіксованість вихідного коду програми і, через це, певну статичність картини системних визовів, що генеруються), можна не навчатися на прикладах, а спробувати, використовуючи вихідний код, задати статичну граматику, що описує деяку мову визовів. У роботі [18] таку граматику задано шляхом аналізу вихідних текстів програм. Запропоновано три варіанти моделей: модель графу визовів (недетермінований скінченний автомат), стекову модель (контекстно вільну граматику) і модель, схожу на описану в роботі [4] (для даної *k*-грами перевіряється, чи належить вона до мови, що породжується певною граматикою). Перевагою такого підходу є відсутність помилок типу false positives через залежність побудованих моделей безпосередньо від вихідного коду програми. Тобто несумісна з моделлю послідовність визовів відразу сигналізує про відхилення роботи програми від її вихідного коду, і тільки такі відхилення сигналізуватимуть про аномалію, а все, що передбачено текстом програми, буде прийнято. Основний недолік подібного підходу — складний процес побудови моделі за аналізом текстів програми. Крім того, є обмеження на види програм, які можуть бути описані статичними граматиками [18].

Модель, якій не властиві недоліки HMM та марківських ланцюгів з фіксованим порядком, була запропонована нами в роботі [19]. Її основна ідея — використовувати модель марківських ланцюгів змінного порядку [20], оскільки наступний символ або сигнал у послідовностях, що генеруються в комп’ютерних системах, рідко визначається попереднім контекстом постійної

довжини. Врахування лише суттєвих контекстів дозволяє обійти експонеційне зростання вимог до ресурсів, не погіршуючи точність моделі. Крім того, більшість попередніх підходів до моделювання поведінки є неадаптивними. Найкраще, що пропонується, — періодичне перенастроювання моделі. Звідси втрати часу і зменшення надійності з часом, що проходить від чергового перетренування. Для підтримки реальної адаптивності ми застосували певну процедуру зміни параметрів нашої моделі так, що остання здобута інформація стає вагомішою [19]. При цьому зберігаються апроксимуючі властивості вихідної моделі. Єдиною відомою нам роботою, де застосований схожий на наш метод адаптивного настроювання ймовірностей, є [13]. Проте там за основу взято найпростішу марківську модель першого порядку, а тому не враховується контекст довжини більшої, ніж одна команда.

Інші методи. Як приклад підходу на основі data-mining технік наведемо роботу [21], де їх застосовують для виявлення з масиву даних асоціативних правил та т.зв. частих епізодів. Асоціативні правила — це вирази виду $[\mathrm{X} \to \mathrm{Y}, \mathrm{c}, \mathrm{s}]$, де X,Y — підмножини множини значень на множині атрибутів, $s = Support(X \cup Y)$ — відсоток записів у базі, що містять $X \cup Y$, $c = \dfrac{Support(X \cup Y)}{Support(X)}$. Тоді правило вигляду $[\mathrm{lynx} \to \mathrm{www.google.com}, 0.2, 0.4]$ означає, що користувач у 40% випадків, коли він користується інтернет браузером `lynx`, заходить на сайт `www.google.com`, і відвідання цього сайту складає 20% від загальної його активності. Автори застосували метод частих епізодів і деякі спеціальні модифікації загальних data-mining технік для пристосування їх для виявлення аномалій. У процесі роботи нові правила додаються, поки множина правил не стабілізується, тобто за вхідними даними вже не можна отримати нових даних, і навчання припиняється.

У роботі [22] для виявлення аномалій використовуються баєсівські мережі — графічні моделі представлення у власній структурі залежностей між об’єктами та розподілів імовірностей. За множиною тренувальних даних оцінюються кореляції між станами і будується мережа із з’єднаними, залежними між собою, вузлами, ймовірності переходів між станами, які

заповнюються, і зв’язані з вузлами таблиці розподілів імовірностей за даними станами вузлів-предків. Через жорстку залежність побудованої структури від наданих тренувальних даних отримана модель не є адаптивною.

Теоретико-інформаційний підхід пропонується в [23], де наводяться два методи вибору оптимальної ширини вікна для виявлення аномалій у послідовностях програмних системних визовів. Перший —обчисленням умовної ентропії $H(X|Y)$ наступного символу $X \in \Sigma$ за умови попереднього контексту $Y \in \Sigma^n$. Те значення n, окреме для кожної програми, за якого значення ентропії мінімальне, вибирається за ширину вікна. Другий метод — використанням ансамблів імовірнісних дерев, аналогічних тим, що використовуються в [19]. Кожне враховується з певною вагою при обчисленні ймовірності наступного системного визову. Якщо якесь дерево правильно "вгадало" наступний визов, його вага в ансамблі збільшується, чим досягається адаптивність.

У роботі [14] описано цікавий підхід, який, на відміну від традиційних методик виявлення аномалій після навчання за неаномальними даними, призначений для виявлення аномалій без попереднього тренування на прикладах нормальної активності. Для цього сукупність послідовних даних представлено як таку, що була згенерована "змішаною" стохастичною моделлю. Тобто з (малою) ймовірністю λ елемент послідовності є аномальним і з ймовірністю $(1-\lambda)$ — нормальним. Розподіл A аномальних елементів за відсутності якихось апріорних даних про нього вважається рівномірним. Розподіл нормальних елементів M може оцінюватися будь-яким методом з арсеналу технік машинного навчання. Отже, маємо генеруючий розподіл-суміш $D = (1 - \lambda)M + \lambda A$. Далі, вважаючи, що відповідні розподіли відомі, можна обчислити функцію правдоподібності $L(D)$ для даної послідовності елементів. Тоді, якщо поточний елемент x_t перемістити з нормального розподілу M в аномальний A і при цьому правдоподібність збільшиться, то елемент там і залишається, якщо ні — нічого не змінюється. Застосування цього підходу дає можливість не залежати від "чистоти" тренувальних даних, які можуть містити і аномальні включення (за умови їх незначної кількості). В роботі наводиться експериментальне порівняння цього методу (де M — марківський ланцюг третього порядку) з двома іншими.

Є спроби використовувати для задачі виявлення аномалій генетичних алгоритмів — алгоритмів пошуку оптимуму, що базуються на аналогіях з природним добором в природі. За "популяцію" береться множина "особин" — бінарних рядків фіксованої довжини. Кількість рядків у множині також фіксована. У процесі еволюції, за певним критерієм — "пристосованістю до навколишнього середовища" вибираються рядки, з яких генерується наступне покоління рядків. Генерація наступного покоління відбувається з використанням трьох основних операторів над рядками: селекція (вибір найпристосованішого представника), репродукція (обмін частинами рядків між собою) і мутація (випадкові зміни бітів у рядках для запобігання незворотної втрати інформації). Так, в [24] генетичні алгоритми використовуються для знаходження песимістичного сценарію в гібридній задачі виявлення зловживань і аномалій. З одного боку, виявляються тільки відомі атаки, з іншого — застосування генетичних алгоритмів і несуворих обмежень на гіпотези може дозволити реєструвати також і варіації цих атак. Нехай N_e і N_a — кількість аудит-подій, що реєструються в системі і кількість відомих атак. Тоді стовпці матриці $\hat{A}_{N_e \times N_a}$ вказують, скільки подій кожного типу є наявними в аудиті. Вектора R, O та H — відповідно, вектор довжини N_a ризиків пов’язаних з атаками, вектор довжини N_e кількостей подій, наявних у поточному аудиті та вектор-гіпотеза з одним одиничним компонентом на місці, яке відповідає певній атаці. Тоді завданням генетичного пошуку є знаходження вектору H, який максимізує добуток (R, H) (тобто атаки, яка максимізує втрати) за умови, що $(\hat{A}H)_i < O_i, 1 \le i \le N_a$. Підхід був перевірений на штучних даних зі змодельованими простими атаками.

Інші огляди. Серед ранніх оглядів варто згадати опис деяких застосувань методів штучного інтелекту до виявлення вторгнень, наведений в [25]. Порівняльний аналіз невичерпної кількості методів виявлення аномалій, що базується на системних визовах, наведений в роботі [26]. Розглядаються приклади чотирьох основних підходів до задачі: просте запам'ятовування підпослідовностей, що трапляються в тренувальних даних, частотні методи, data-mining і HMM. Всі конкретні моделі було перевірено на великих масивах реальних даних як з аномальними включеннями, так і без них, і зроблено висновок, що HMM-модель є найкращою, хоча й не найефективнішою.

У нашому огляді ми в основному розглядали математичний аспект проблеми моделювання поведінки суб'єктів комп'ютерної системи. Важливим є також питання порівняльної характеристики існуючих IDS та їхньої класифікації. Так, в огляді [27] розглянуто 20 (від перших до тих, що з'явилися порівняно недавно) IDS і вміщено їхню докладну класифікацію за багатьма ознаками. В огляді [28] детально описуються дослідницькі, комерційні і доступні для широкої публіки системи виявлення вторгнень. Також перераховано проблеми, які мають розв'язати майбутні IDS, і надано рекомендації з вибору, підтримки цих систем.

Висновки

Ми розглянули основні моделі, що застосовуються в ADS, проте жодна з них не гарантує виявлення всіх атак, тому в реальних IDS ми рекомендуємо користуватися сукупністю показників, отриманих за різними моделями і за ними робити остаточний висновок про наявність чи відсутність вторгнень, їх характер тощо.

Моделі не повинні залежати від конкретного типу аудит-даних, тоді їх можна застосувати до будь-яких аудит-послідовностей. Так, такими даними, окрім команд, що безпосередньо виконуються користувачем, можуть стати послідовності системних визовів, характерні для даної програми, значення інтервалів між натисканнями клавіш на клавіатурі, інтервали між робочими сесіями, послідовні значення координат положення курсору миші на екрані, послідовності значень полів заголовку мережевих пакетів усіх рівнів або величини, що в них не містяться, але від них

залежать тощо. Тобто будь-яка послідовність сигналів, команд, елементів або хронологічних чи інших значень, характерна для користувача, процесу, системи або мережевого сегмента, може бути використана в ADS.

Користувачам властиво змінювати свою поведінку (через зміну задач, набуття нових звичок тощо), тоду моделі обов'язково повинні бути адаптивними.

Велика частина атак є атаками проти системних програм, наприклад набуття прав суперкористувача шляхом використання вразливостей у SUID програм. Після цього програма, як правило демонструє кардинально відмінний характер системних визовів від тих, що спостерігалися до атаки [3,4,10-12,14,23,26]. Тому слід звернути увагу на виявлення аномалій на рівні системних визовів, який стає дедалі популярнішим, оскільки дає змогу абстрагуватися від нерегулярної людської поведінки. Одночасно не можна відмовлятися від аналізу аудит-даних, що поступають від користувачів, оскільки тільки аналіз на цьому рівні уможливлює виявлення вторгнень, які на рівні системних визовів не проявляються (наприклад, цілком законне з точки зору підсистеми аутентифікації використання вкраденого пароля).

Повноцінна IDS повинна включати також і компоненту з виявлення зловживань, оскільки на сьогоднішній день атаки з використанням експлойтів залишаються одними з основних видів атак.

1. S. Axelsson. The base-rate fallacy and its implications for the difficulty of intrusion detection. In ACM Conference on Computer and Communications Security, pages 1-7, 1999.
2. D. E. Denning. An intrusion-detection model. In Proc. IEEE Symposium on Security and Privacy, pages 118-131, 1986.
3. A. Somayaji. Automated response using system-call delays. In Proc. of USENIX Security Symposium 2000, pages 185-197, 2000.
4. S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for Unix processes. In Proc. of the 1996 IEEE Symposium on Research in Security and Privacy, pages 120-128. IEEE Computer Society Press, 1996.
5. T. Lane and C. E. Brodley. Temporal sequence learning and data reduction for anomaly detection.

ACM Transactions on Information and System Security, 2(3): 295-331, 1999.

6. M. Theus and M. Schonlau. Intrusion detection based on structural zeroes. Statistical Computing and Graphics Newsletter, 9(1): 12-17, 1998.
7. K. Tan. The application of neural networks to unix computer security. In Proc. of the IEEE International Conference on Neural Networks, volume 1, pages 476-481, Perth, Australia, 1995.
8. J. Ryan, M. Lin, and R. Miikkulainen. Intrusion detection with neural networks. Advances in Neural Information Processing Systems, pages 254-272, 1998.
9. А. М. Резник, Н. Н. Куссуль, А. М. Соколов. Нейросетевая идентификация поведения пользователей компьютерных систем // Кибернетика и вычислительная техника, 1999. – Вып. 123. – С. 70-79.
10. D. Endler. Intrusion detection: Applying machine learning to solaris audit data. In Proc. Annual Computer Security Applications Conference (ACSAC'98), pages 268-279, Los Alamitos, CA, Dec. 1998. IEEE Computer Society Press. Scottsdale, AZ.
11. Ghosh, J. Wanken, and F. Charron. Detecting anomalous and unknown intrusions against programs. In Proc. of the 1998 Annual Computer Security Applications Conference (ACSAC'98), Dec. 1998. http://rstcorp.com/~anup/ACSAC98.pdf
12. K. Ghosh, A. Schwartzbard, and M. Schatz. Learning program behavior profiles for intrusion detection. In Proc. 1-st USENIX Workshop on Intrusion Detection and Network Monitoring, pages 51-62, Santa Clara, California, Apr. 1999.
13. D. Davison and H. Hirsh. Predicting sequences of user actions. In Predicting the Future: AI Approaches to Time-Series Problems, pages 5-12, Madison, WI, July 1998. AAAI Press. Proc. of AAAI-98/ICML-98 Workshop.
14. E. Eskin. Anomaly detection over noisy data using learned probability distributions. In Proc. 17th International Conf. on Machine Learning, pages 255-262. Morgan Kaufmann, San Francisco, CA, 2000.
15. N. Ye. A markov chain model of temporal behavior for anomaly detection. In Proc. of the 2000 IEEE

Systems, Man, and Cybernetics, Information Assurance and Security Workshop, 2000. http://www.itoc.usma.edu/marin/Wshop/Papers2000/WA1_1.pdf

16. T. Lane. Hidden markov models for human/computer interface modeling. In IJCAI-99 Workshop on Learning About Users, pages 35-44, 1999.
17. C. C. Michael and A. Ghosh. Two state-based approaches to program-based anomaly detection. ACM Transactions on Information and System Security, 5(2), 2002. http://www.acsac.org/2000/papers/96.pdf
18. D. Wagner and R. Dean. Intrusion detection via static analysis. In Proc. of the 2001 IEEE Symposium on Security and Privacy, pages 156-169, Los Alamitos, CA, May 14-16 2001.
19. Н. Н. Куссуль, А. М. Соколов. Адаптивное обнаружение аномалий в поведении пользователей компьютерных систем с помощью марковских цепей изменяющегося порядка // Кибернетика и вычислительная техника (подано в печать).
20. D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. Machine Learning, 25(2-3): 117-149, 1996.
21. W. Lee and S. Stolfo. Data mining approaches for intrusion detection. In Proc. of the 7th USENIX Security Symposium, pages 79-94, San Antonio, TX, 1998.
22. N. Ye, M. Xu, and S. M. Emran. Probabilistic networks with undirected links for anomaly detection. In Proc. of the 2000 IEEE Workshop on Information Assurance and Security, West Point, NY, June 2000. http://www.itoc.usma.edu/marin/Wshop/Abstracts/WA1_2.pdf
23. E. Eskin, W. Lee, and S. J. Stolfo. Modeling system calls for intrusion detection with dynamic window sizes. In Proc. of DARPA Information Survivability Conference and Exposition (DISCEX II), June 2001. http://www.cs.columbia.edu/ids/publications/smt-syscall-discex01.pdf
24. L. Me. GASSATA, a genetic algorithm as an alternative tool for security audit trails analysis. In First international workshop on the Recent Advances in Intrusion Detection, Louvain-la-Neuve, Belgium, Sept. 1998. http://www.raid-symposium.org/raid98/Prog_RAID98/Full_Papers/gassata.pdf
25. J. Frank. Artificial intelligence and intrusion detection: Current and future directions. In Proc. of the

17th National Computer Security Conference, pages 22-33, Baltimore, MD, 1994.

26. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: Alternative data models. In Proc. of the 1999 IEEE Symposium on Security and Privacy (SSP '99), pages 133-145, May 1999. IEEE.
27. J. Allen et al. State of the practice of intrusion detection technologies. TR CMU/SEI-99-TR-028, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, Jan. 2000.
28. S. Axelsson. Research in intrusion-detection systems: A survey. TR 98-17 SE-412 96, Department of Computer Engineering, Chalmers University of Technology, Geteborg, Sweden, Dec. 1998.