ОГЛЯД СИСТЕМ ВИЯВЛЕННЯ ВТОРГНЕНЬ НА ОСНОВІ HONEYPOT-ТЕХНОЛОГІЙ

Владислава Волянська, Андрій Гізун, Віктор Гнатюк

Національний авіаційний університет

ВОЛЯНСЬКА Владислава Вікторівна

Рік та місце народження: 1977 рік, м. Плавськ, Росія.
Освіта: Національний авіаційний університет, 2004 рік.
Посада: IT-Manager for network infrastructure, Apogeum Sp. z o.o. Poland, здобувач кафедри безпеки інформаційних технологій.
Наукові інтереси: інформаційна безпека операційних систем, управління інцидентами інформаційної безпеки, комплексні системи захисту інформації.
Публікації: більше 15 наукових публікацій, серед яких наукові статті, матеріали і тези доповідей на конференціях.
E-mail: volyanska.vladyslava@gmail.com

ГІЗУН Андрій Іванович

Рік та місце народження: 1987 рік, м. Нетішин, Хмельницька область, Україна.
Освіта: Національний авіаційний університет, 2010 рік.
Посада: асистент кафедри безпеки інформаційних технологій з 2012 року.
Наукові інтереси: інформаційна безпека, управління інцидентами інформаційної безпеки, штучні імунні системи, управління безперервністю бізнесу та правове забезпечення захисту інформації.
Публікації: більше 15 наукових публікацій, серед яких наукові статті, матеріали і тези доповідей на конференціях, авторські свідоцтва.
E-mail: caesar07@meta.ua

ГНАТЮК Віктор Олександрович

Рік та місце народження: 1990 рік, м. Нетішин, Хмельницька область, Україна.
Освіта: Хмельницький національний університет, 2012 рік.
Посада: аспірант кафедри безпеки інформаційних технологій з 2012 року.
Наукові інтереси: інформаційна безпека, управління інцидентами інформаційної безпеки.
Публікації: 5 наукових публікацій, серед яких наукові статті, тези та матеріали доповідей на конференціях.
E-mail: viktorgnatyuk@meta.ua

Анотація. *У цій статті проведено огляд існуючих систем виявлення та попередження вторгнень на базі honeypot-технологій. Також, наведено порівняльну характеристику методів генерації сигнатур систем виявлення вторгнень з використанням віртуальних приманок. Ці результати будуть корисними при розробці нових та удосконаленні існуючих систем виявлення вторгнень.*

Ключові слова: *система виявлення вторгнень, віртуальна приманка, honeypot, механізм виявлення атаки, сигнатура, мережевий трафік.*

Вступ

Збільшення розміру і охоплення мережі Інтернет, що призводить до генерації величезного об'єму мережевого трафіку, підвищує необхідність у виявленні комп'ютерних вторгнень. Крім того ставляться вимоги удосконалення наявних систем виявлення вторгнень (СВВ) щодо їх продуктивності і оперативності. Ефективна реалізація СВВ повинна відрізняти основний клієнтський трафік від спроб атак, одночасно справляючись з проблемами пропускної здатності, затримки і безпеки мережі.

Забезпечення зазначеної концепції ефективно реалізується за допомогою так званих гібридних систем [1]. Останні включають в себе низько-рівневі віртуальні приманки – як первинний сенсор і джерело вхідної інформації для генерації сигнатур; високо-рівневі – як середовище стримуванні і подальшого детального аналізу; та мережеву СВВ – як механізм зворотної реакції. Ефективне

стримування кібератак включає 4 послідовних кроки: 1) Фіксація атаки на декількох різнорівневих сенсорах; 2) Генерація сигнатури виявленої атаки; 3) Застосування даної сигнатури СВВ і її розповсюдження; 4) Подальший глибокий аналіз механізму атаки. Така система охоплює усі зазначені компоненти. Як середовище основного аналізу застосовується концепція Honeynet GenIII [2]. Honeypot-приманки реалізуються як віртуальні Windows-машини в середовищі Vmware (їх кількість залежить від продуктивності основної ОС). Кожна з них комплектується набором сенсорів, що виконують всеохоплюючий моніторинг від активності в мережі (Wireshark/Ethereal), до активності системи і користувача: включаючи контроль файлової системи (Filemon, Forensic Toolkit, BinText, ADSScan, IntegCheck), доступу до реєстру (Regmon, RegistryProt), запущених процесів (Process Explorer, Advanced Process Manipulation) та аудит системних подій. Відповідно до вимог Honeynet GenIII [2] в Vmware також реалізується технологія прозорого шлюзу-файрволу для контролю вхідного/вихідного трафіку на базі Unix-системи (можливі реалізації: FreeBSD, OpenBSD, Red Hat Linux) [3]. На шлюз-файрвол встановлюється відкрита СВВ Snort, що може працювати у 2 режимах: як власне СВВ так і простого аналізатора пакетів [4]. Для візуалізації даних Snort, зокрема, використовується веб-інтерфейс ACID (Analysis Console for Intrusion Database) [4] з використанням БД MySQL. Крім того як первинний сенсор на Unix-системі функціонуватиме низько-рівнева віртуальна приманка Honeytrap [5], на базі якої за допомогою модуля Nebula [6], генеруватимуться сигнатури атак у форматі для Snort.

Основною **метою** цієї статті є порівняльний аналіз існуючих систем виявлення вторгнень на базі honeypot-технологій, визначення базових принципів їх побудови, основних переваг і недоліків.

Основна частина дослідження

Перед описом архітектури і основних складових компонентів запропонованої в роботі СВВ на основі віртуальних приманок слід представити короткий огляд існуючих систем, що використовують схожі принципи.

В статті розглянуто більше 10-ти сучасних СВВ. Кожну систему було детально проаналізовано, зокрема особливості її функціонування. Так отримані дані стосовно механізму виявлення атаки та необхідних для виявлення вхідних даних представлені в табл. 1, а особливості процесу формування сигнатур СВВ висвітлені в табл. 2.

Механізм генерації сигнатур системи **Honeycomb** [7] ґрунтуєтьсь на техніці виявлення образів: весь вхідний мережевий трафік порівнюється з трафіком, попередньо фіксованим на віртуальній приманці. Для цього Honeycomb зберігає реасембльований вміст пакетів UDP- та TCP-з'єднань. Після отримання пакета honeypot-системою генерується порожня сигнатура. Далі проводиться аналіз протоколу на мережевому і транспортному рівнях. Виявлені аномалії заголовка записуються у сигнатуру. Після цього отриманий пакет відноситься до вже наявного потоку/з'єднання з однаковими IP-адресами джерела/призначення та номера порту. І нарешті, це з'єднання порівнюється з наявними іншими. Порівняння і виділення спільної частини проводиться на основі алгоритму найдовшого загального підрядка (longest common substring, LCS). Таким чином Honeycomb генерує сигнатури мережевого рівня, що представляють собою шаблони у вигляді неперервної послідовності байтів з можливістю подальшого аналізу протоколів.

Polygraph [8] – система, розроблена для генерації сигнатур поліморфних «хробаків». Класифікатор потоку реасемблює його (відносно порту) і поміщає в пул підозрілого потоку. Саме для нього генеруються сигнатури. Дана система оперує трьома типами сигнатур, що формуються з підрядків, які називаються маркерами (токенами, tokens). Сигнатури сполучення (conjunction signatures) складаються з набору маркерів і відповідають вмісту пакету, якщо всі маркери присутні в ньому в будь-якому порядку. Сигнатури підпослідовності маркерів (Token-subsequence signatures) складаються з впорядкованого набору маркерів відповідають вмісту пакету, якщо в ньому присутні ці маркери в такому ж порядку. Сигнатури Байєса (Bayes signatures) складаються з набору маркерів, кожному з яких поставлено у відповідність певне значення, та загальний поріг. Сигнатура відповідає вмісту пакету, якщо сума значень усіх маркерів перевищує поріг.

Для кожного вхідного пакету система **Earlybird** [9] розраховує варіант відбитку Рабіна (Rabin fingerprints) для всіх можливих підрядків певної довжини. Кожен з таких відбитків хешується разом з портом призначення і протоколом. Значення цих хешів зберігається в таблиці поширеності. Кожного її запису відповідає лічильник, список унікальних джерел і призначень. Таке рішення представляє собою своєрідне просіювання вмісту (content sifting), так як сортування таблиці по відлікам підрядків і розміру списку адрес дає орієнтовний набір підозрілого трафіку мережевих «хробаків». В кінці, система генерує шаблонні сигнатури формату IDS Snort-inline.

Система **Nemean** [10] складається з двох компонентів: Компонента Виділення Даних (the Data Abstraction Component) і Компонента Генерації Сигнатур (the Signature Generation Component). Пакети поступають на Компонент Виділення Даних і нормалізуються. Далі відбувається агрегація потоку: пакети складаються в з'єднання і сесії. Вихід даного компонента – напів-структуроване дерево сесій – використовується як вхід для Компонента Генерації Сигнатур. Перший крок – групування сесій і з'єднань зі схожими профілями атак відповідно до показників подібності. Як алгоритм кластеризації використовується On-line star clustering та два різні показника подібності: косинусоїдальна подібність (cosine similarity) і ієрархічна перевірочна відстань (hierarchical edit distance). Далі використовується навчання автоматів (automata learning) для побудови сигнатури атаки з кластерів сесій та з'єднань.

Система **Autograph** [11] використовує евристику механізму виявлення сканерів портів для розділення трафіку на підозрілий і непідозрілий (розглядаються неуспішні вхідні TCP-підключення). Всі потоки в пулі підозрілих потоків сортуються відповідно до порту призначення. Процес генерації сигнатур ініціюється, коли пул підозрілих потоків містить кількість потоків для конкретного порту призначення більшу, ніж деяке порогове значення. При цьому Autograph визначає частоту появи підрядків, що не перекриваються, серед вмісту усіх підозрілих потоків, і пропонує найчастіший з них на кандидата в сигнатури. Для цього вміст кожного потоку розділяється на блоки різної довжини, використовуючи механізм COntent-based Payload Partitioning (COPP), і підраховується число потоків, що містять кожен з таких блоків. Сигнатури представляються в форматі IDS Bro.

PADS [12] (Position-aware distribution signatures) використовує подвійну honeypot-систему для відслідковування підозрілої активності в локальних мережах. Всі з'єднання, ініційовані вхідною високо-рівневою приманкою, перенаправляються до низько-рівневої вихідної. Головна ідея – це те, що атакована система рано чи пізно починає інфікувати інші. При цьому різні варіанти шкідливого коду будуть легко фіксуватися низько-рівневою приманкою. Сигнатури PADS включають частотний розподіл байтів (а не якесь фіксоване значення) для кожної позиції в рядку сигнатури. Для цього використовуються алгоритм максимізації очікування (expectation-maximization, EM) і вибірковий алгоритм Гіббса.

PAYL [13] – це сенсор виявлення аномалій, що фіксує аномальний вхідний трафік і зрівнює його з вихідним на тому ж порту. Аргументація авторів наступна: якщо аномальні вихідні і вхідні пакети одного порту дуже схожі, то це свідчить про високу ймовірність активності зловмисника чи «хробака». Як показник порівняння використовуються алгоритми найдовшого загального підрядка (LCS) та найдовшої загальної підпослідовності. Ідентифіковані таким чином послідовності байтів можуть бути використані як сигнатури. І нарешті, всі вихідні пакети, що мають показник схожості вищий деякого порогового значення, блокуються для недопущення поширення атаки.

Таблиця 1. Порівняльна характеристика СВВ з використанням віртуальних приманок

СВВ	**Механізм виявлення атак**	**Вхідні дані для виявлення атак**
Autograph	Виявлення на основі сканування	Трафік мережі
PADS	Перенаправлення вихідних з'єднань	Трафік мережі
PAYL	Машинне навчання частотного розподілення байтів	Трафік мережі
COVERS	Рандомізації Адресного Простору ASR	Інформація про доступ до пам'яті
DIRA	Внесення датчиків у вихідний код	Розміщення в пам'яті адрес повернення та вказівок на функції
DOME	Розбіжності зі статичною моделлю системних викликів	Інформація про виконувані системні виклики
Minos	Динамічний аналіз зараження	Операції читання/запису в пам'яті
Paid	Розбіжності зі статичною моделлю системних викликів	Інформація про виконувані системні виклики
Vigilante	Динамічний аналіз зараження	Операції читання/запису в пам'яті
HoneyStat	Фіксація вихідних з'єднання, змін ФС та стану стеку	Логи ФС та стеку, інформація про вихідні з'єднання

Таблиця 2. Порівняльна характеристика генерації сигнатур СВВ з використанням віртуальних приманок

СВВ	**Вхідні дані для генерації сигнатур**	**Тип генерованих сигнатур**	
		Рівень представлення	**Форма представлення**
Honeycomb	Реасембльований вміст потоку	Мережевий	Неперервна послідовність байтів
Polygraph	Реасембльований вміст потоку	Мережевий	Імовірнісний розподіл байтів
Earlybird	Вміст окремого пакету	Мережевий	Неперервна послідовність байтов (Snort)
Nemean	Реасембльований вміст нормалізованих пакетів	Мережевий	Кінцевий автомат
Autograph	Реасембльований вміст пакетів	Мережевий	Неперервна послідовність байтов (Bro)
PAYL	Вміст окремого пакету	Мережевий	Частотний розподіл байтів
COVERS	Вміст окремого пакету	Мережевий	Характеристики типу/поля повідомлення
DOME	Двійковий код програми	Хост	Модель використання системних викликів програмою
Paid	Вихідний код	Хост	Модель використання системних викликів програмою
Vigilante	Операції читання/запису в пам'яті	Мережевий/ Хост	Весь трафік, необхідний для спрацювання детектора, плюс специфічна інформація про атаку

Система **COVERS** [14] складається з частини виявлення атаки і частини генерації сигнатури. Виявленн атаки реалізується на основі техніки Рандомізації Адресного Простору (Address Space Randomization, ASR). ASR використовується для випадкової зміни розміщення об'єктів в пам'яті. При цьому зловмисник повинен вгадати, де знаходиться необхідна йому частина коду. Якщо він помиляється, генерується заборона доступу до пам'яті. Далі визначається мережевий пакет чи потік, задіяний в атаці, і послідовність байтів в цьому пакеті, що спричинили тривогу.

Система **DIRA** [15] (The attack Detection, Identification and Repair) – розширення GCC, що дозволяє компільованим програмам виявляти і ідентифікувати атаки перехоплення контролю над потоком (control-flow hijacking). Це реалізується шляхом внесення під час рекомпіляції датчиків (структур даних чутливих до контролю), що можуть виявити недозволені зміни. Ідентифікація атаки складається з відслідковування вхідних даних, що спричинили тривогу. Це виконується за допомогою оновлюваного логу пам'яті глобальних і статичних змінних та транзакцій. Заключний етап – відкат стану програми до точки, яка передувала зчитування входу, що містив атаку, та перезавантаження звідти програми.

Рішення **DOME** [16] (The Detection Of Malicious Executables) дозволяє виявити атаки проникнення і атаки ініційовані виконавчими файлами (exe), що модифіковані за допомогою заплутаного коду або містять динамічно генерований код. Механізм виявлення ґрунтується на тому, що шкідливий код часто використовує системні виклики, проникає в запущену програму і намагається приховати свою присутність. DOME проводить статичний аналіз (за допомогою IDA Pro disassembler) виконуючих файлів для визначення розміщення системних викликів, і далі слідкує чи змінюється їх розміщення під час роботи програми. Хоча DOME і не генерує сигнатур шкідливої активності, система може розпізнати таку активність, порівнюючи її з нормальною /підтвердженою поведінкою.

Система **Minos** [17] являє собою модифікацію архітектури Pentium ядра ОС, що дозволяє зупинити атаки на контрольні дані, шляхом помічування неперевірених вхідних даних як заражених, та їх поширення операціями ФС, пам'яті і процесора. Це представляє собою динамічний аналіз зараження (dynamic taint analysis).

Система **Paid** [18] (The Program semantics-Aware Intrusion Detection system) захищає програми від атак перехоплення контролю над потоком з використанням системних викликів. Вона використовує вихідний код програми для створення моделі системних викликів, що включає наступні три характеристики: розміщення і порядок слідування викликів та частин контрольного потоку програми. Виявлення аномальної поведінки відбувається через порівняння зразків реального функціонування з даною моделлю.

Система **Vigilante** [19] включає компонент генерації фільтрів для захисту кінечних хостів від послідовних атак та сполученої мережі для швидкого розповсюдження тривоги. Головна особливість Vigilante – концепція само засвідчуваних тривог (self-certifying alerts, SCA). Система складається з наступних компонентів: детектор, контролер SCA, генератор фільтрів та розповсюджувач попереджень.

HoneyStat [20] емулює кілька ОС за допомогою VMware GSX Server. Система виявляє 3 різних типи подій: події в пам'яті, мережі і дисковому просторі. Події в пам'яті фіксуються за допомогою ПЗ виявлення переповнення буферу на віртуальній приманці. Мережева подія створюється у випадку генерації приманкою вихідного трафіку. Події дискового простору виникають, якщо процес записує інформацію у задані частини ФС. Таким чином інформація, що фіксується HoneyStat включає: дані про ОС хоста, а також тип подій і відповідні зафіксовані дані (події пам'яті – стан стеку, події мережі – вихідні пакети, події дискового простору – зміни файлів).

Висновки

Таким чином, у цій роботі проведено огляд існуючих систем виявлення на базі honeypot-технологій, а також проведено порівняльний аналіз методів генерації сигнатур систем виявлення вторгнень з використанням віртуальних приманок. У майбутньому отримані в роботі результати будуть використані при розробці нових та удосконаленні існуючих систем виявлення вторгнень. Це буде корисним, в першу чергу, командам реагування на інциденти інформаційної безпеки та різного роду кризовим центрам.

Література

[1] Kyaw K.L.L. Hybrid Honeypot System for Network Security / K.L.L. Kyaw, P. Gyi // World Academy of Science, Engineering and Technology 48
2008. – Mandalay: Mandalay Technological University, 2008.

[2] Balas E. Honeynet data analysis: A technique for correlating sebek and network data / E. Balas // Workshop on Information Assurance and Security United States Military Academy, West Point, NY. – IEEE, 2004.

[3] Hope P. Mastering FreeBSD and OpenBSD Security / P. Hope, Y. Korff, B. Potter. – Ca.: O'Reily Media, 2005. – P.464.

[4] Cox K., Gerg C. Managing Security with Snort and IDS Tools / K. Cox, C. Gerg. – Ca.: O'Reily Media, 2004. – p.288.

[5] Honeytrap – A Dynamic Meta-Honeypot Daemon *[Електронний ресурс]: (honeytrap)* – Режим доступу: http://honeytrap.carnivore.it/

[6] Nebula – Generating Syntactical Network Intrusion Signatures / Werner T., Fuchs C., Gerhards-Padilla E., Martini P. // Lecture Notes in Computer Science. – B.: Springer Berlin, 2005. – Vol.3245. – P.105-113.

[7] Kreibich C. Honeycomb - creating intrusion detection signatures using honeypots / C. Kreibich, J. Crowcroft // Second Workshop on Hot Topics in Networks (Hotnets II). – Boston, 2003.

[8] Newsome J. Polygraph: Automatically generating signatures for polymorphic worms / J. Newsome, B. Karp, D. Song // Proceedings of the 2005 IEEE Symposium on Security and Privacy. – Washington: IEEE Computer Society, 2005.

[9] Automated worm fingerprinting / Singh S., Estan C., Varghese G., Savage S. // OSDI. – San Diego.: University of California, 2004. – P.45-60.

[10] Yegneswaran V. An architecture for generating semantic-aware signatures / V. Yegneswaran, J.T. Giffin, S.J. Paul Barford // Proceedings of the 14th USENIX Security Symposium, 2005. – P.97-112.

[11] Kim H.A. Autograph: Toward automated, distributed worm signature detection / H.A. Kim, B. Karp // Proceedings of the USENIX Security Symposium, 2004. – P.271-286.

[12] Tang Y. Defending against internet worms: A signature-based approach / Y. Tang, S. Chen // Proceedings of IEEE INFOCOM'05, 2005.

[13] Wang S.S.K. Anomalous payload-based network intrusion detection / S.S.K. Wang // Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection. – Sophia Antipolis, 2004.

[14] Liang Z. Fast and automated generation of attack signatures: A basis for building self-protecting servers / Z. Liang, R. Sekar // Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS). – Alexandria, 2005.

[15] Smirnov C.T. Dira: Automatic detection, identification, and repair of control-hijacking attacks. / C.T. Smirnov // Proceedings of NDSS05: Network and Distributed System Security Symposium Conference Proceedings. – San Diego, 2005.

[16] Detection of injected, dynamically generated, and obfuscated malicious code / Rabek J.C., Khazan R.I., Lewandowski S.M., Cunningham R.K. // Proceedings of the 2003 ACM workshop on Rapid Malcode. – New York, 2003. – P.76-82.

[17] Crandall J.R. Experiences using minos as a tool for capturing and analyzing novel worms for unknown vulnerabilities / J.R. Crandall, S.F. Wu, F.T. Chong // Proceedings of DIMVA'05, 2005. – P.32-50.

[18] Lam L., Chiueh T. Automatic extraction of accurate application-specific sandboxing policy / L. Lam, T. Chiueh // Lecture Notes in Computer Science. – B.: Springer Berlin, 2004. – Vol.3224. – P.1-20.

[19] Security for structured peer-to-peer overlay networks / Castro M., Druschel P., Ganesh A., Rowstron A., and Wallach D.S.// Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI'02). – Boston, 2002.

[20] Honeystat: Local worm detection using honeypots / Dagon D., Qin X., Gu G., Lee W., Grizzard J.B., Levine J.G., and Owen H.L. // Lecture Notes in Computer Science. – B.: Springer Berlin, 2004. – Vol.3224. – P. 39-58.

УДК 004.056.53:004.492.3 (045)

Волянская В.В., Гизун А.И., Гнатюк В.А. Обзор систем обнаружения вторжений на основе honeypot-технологий

Аннотация. *В статье проведен обзор существующих систем выявления и предупреждения вторжений на базе honeypot-технологий. Также приведено сравнительную характеристику методов генерации сигнатур систем выявления вторжений с использованием виртуальных приманок. Полученные результаты будут полезными при разработке новых и совершенствовании существующих систем выявления вторжений.*

Ключевые слова: *система выявления вторжений, виртуальная приманка, honeypot, механизм выявления атаки, сигнатура, сетевой трафик.*

Volyanska V.V., Gizun A.I., Gnatyuk V.O. Review of intrusion detection systems based on honeypot technology

Abstract. *In the paper the review of intrusion detection systems based on honeypot technology was carried out. The comparative description of the generation methods for signatures of intrusion detection systems based on virtual honeypot also was given. These results will be useful for development of new and improvement of existed intrusion detection systems.*

Keywords: *intrusion detection system, virtual honeypot, honeypot, attack detection mechanism, signature, network traffic.*

Отримано *10 жовтня 2012 року*, затверджено редколегією *22 листопада 2012 року*
(рецензент д.т.н., професор О.Г. Корченко)