Лекція 1.

1. Класифікація комп’ютерних вірусів.
2. Історія вірусів.

1. Класифікація комп’ютерних вірусів.

Вірус — програма, яка має 2 ознаки:
— здатність до самовідтворення;
— нанесення шкоди (захаращення пам’яті та місця на диску, поглинання інформації, зниження продуктивності роботи користувачів).

Комп’ютерні віруси — різновид шкідливих програм, характерною особливістю яких є **здатність до розмноження (самореплікації)**. На додачу до цього, вони *можуть* пошкоджувати чи повністю знищувати дані, підконтрольні користувачу, від імені якого була запущена заражена програма.

Неспеціалісти інколи відносять до комп’ютерних вірусів усі шкідливі програми, такі як мережні черв’яки, троянські коні та програми-шпигуни.

Віруси розповсюджуються, вписуючи себе у виконуваний код інших програм.

Слід зазначити, що, будучи програмою, вірус може заразити лише програму. Будь-які зміни непрограм є не зараженням, а просто пошкодженням даних, оскільки така копія вірусу ніколи не одержить керування, бо не використовуватиметься процесором в якості інструкцій (що вже не відповідає сутності вірусу). Саме тому неформатований (plain) текст не може бути переносником вірусу.

З огляду на вищесказане, довгий час вважалося, що текст взагалі неможливо заразити. Але з появою документів, що містили макроси, це твердження довелося переглянути в тому плані, що саме можна вважати текстом. Виходить, що документ MS Word — не текст, а програма (чи текст з програмою), яка може бути заражена.

Три умови існування шкідливих програм

Операційна система або прикладна програма може піддатися вірусному нападу в тому випадку, якщо вона має можливість запустити програму, що не є частиною самої системи. Даній умові задовольняють усі популярні "настільні" операційні системи, багато офісних програм, графічні редактори, системи проектування та інші програмні комплекси, що мають убудовані мови сценаріїв (скриптів).

Комп'ютерні віруси, черв’яки, троянські програми існують для десятків операційних систем і додатків. У той же час існує величезна кількість інших операційних систем і додатків, для яких шкідливі програми поки не виявлені. Що є причиною існування шкідливих програм в одних системах і його відсутності в інших?

Причиною появи подібних програм у конкретній операційній системі або додатку є одночасне виконання наступних умов:

— популярність, широке поширення даної системи;
— наявність різноманітної та достатньо повної документації по системі;
— незахищеність системи або існування відомих вразливостей у системі безпеки.

Кожна перерахована умова є необхідною, а виконання всіх трьох умов одночасно є достатнім для появи різноманітних шкідливих програм.

Умова популярності системи необхідна для того, щоб вона попалася на очі хоча б одному комп'ютерному хуліганові або гакеру. Якщо система існує в одиничних екземплярах, то ймовірність її зловмисного використання близька до нуля. Якщо ж виробник системи домігся її масового поширення, то очевидно, що рано або пізно гакери і вірусописці спробують використати її в своїх інтересах.

Напрошується природний висновок: чим популярніші операційна система або додаток, тим частіше вони будуть жертвами вірусної атаки. Практика це підтверджує — розподіл кількості шкідливого програмного забезпечення для Windows і Linux практично збігається із частками ринку, які займають ці операційні системи.

Наявність повної документації необхідна для існування вірусів із природної причини — створення програм (включаючи вірусні) неможливе без технічного опису використання сервісів операційної системи й правил написання додатків. У більшості мобільних телефонів, наприклад, подібна інформація закрита — ані компанії-виробники програмних продуктів, ані гакери не мають можливості розробляти програми для даних пристроїв. У деяких "розумних" телефонів є документація для розробки додатків — і, як наслідок, з'являються й шкідливі програми, розроблені спеціально для телефонів даного типу.

Під захищеністю системи розуміють архітектурні рішення, які не дозволяють новому (невідомому) застосуванню отримати повний або достатньо широкий доступ до файлів на диску (включаючи інші застосування) і потенційно небезпечним сервісам системи. Подібне обмеження фактично блокує будь-яку вірусну активність, але при цьому, природно, накладає істотні обмеження на можливості звичайних програм. Прикладів широко відомих захищених операційних систем і застосувань, на жаль, немає. Частково задовольняє вимозі захищеності Java-машина, яка запускає Java-застосування в режимі «пісочниці». І дійсно, «справжніх» вірусів і троянських програм у вигляді Java-застосувань не було достатньо довгий час (за винятком тестових вірусів, які були практично непрацездатні). Шкідливі програми у вигляді Java-застосувань з'явилися лише тоді, коли були виявлені способи обходу вбудованої в Java-машину системи безпеки.

Класифікація

Спеціалісти „Лабораторії Касперського" пропонують поділяти шкідливі програми на три великі класи:

1. TrojWare: троянські програми, нездатні до самостійного розмноження (backdoor, rootkit і різні trojan);
2. VirWare: здатні до самостійного розмноження (віруси та черв'яки);
3. Other MalWare: програмне забезпечення, що інтенсивно використовується зловмисниками для створення шкідливих програм і організації атак.

Щодо вірусів, то в даний час не існує єдиної системи класифікації і іменування вірусів (хоча спроба створити стандарт була зроблена на зустрічі CARO (Computer Antivirus Research Organization) в 1991 році). Прийнято розділяти віруси за об'єктами, що вражаються (файлові віруси, завантажувальні віруси, скриптові віруси, мережні черв'яки), за операційними системами й платформами, що вражаються (DOS, Windows, Unix, Linux, Java та інші), за технологіями, використовуваними вірусом (поліморфні віруси, стелс-віруси), за мовою, якою написано вірус (асемблер, високорівнева мова програмування, скриптова мова та ін.).

Класичні віруси

Типи комп'ютерних вірусів розрізняються між собою по наступних основних ознаках:

— середовище існування;
— спосіб зараження.

Під «середовищем існування» розуміються системні області комп'ютера, операційні системи або застосування, в компоненти (файли) яких упроваджується код вірусу. Під «способом зараження» розуміються різні методи впровадження вірусного коду в об'єкти, що заражаються.

Середовище існування

За середовищем існування віруси можна розділити на:

— файлові;
— завантажувальні;
— макро;
— скриптові.

Файлові віруси при своєму розмноженні тим або іншим способом використовують файлову систему якої-небудь (або яких-небудь) ОС. Вони:

різними способами упроваджуються у виконувані файли (найбільш поширений тип вірусів);
створюють файли-двійники (компаньон-віруси);
створюють свої копії в різних каталогах;
використовують особливості організації файлової системи (link-віруси).

Завантажувальні віруси записують себе або в завантажувальний сектор диска (boot-сектор), або в сектор, системний завантажувач вінчестера (Master Boot Record), що містить, або міняють покажчик на активний boot-сектор. Даний тип вірусів був достатньо поширений в 1990-х, але практично зник з переходом на 32-бітові операційні системи і відмовою від використання дискет як основного способу обміну інформацією.

Але в 2005 році спеціалісти з eEye Digital Security в своїй доповіді на конференції Black Hat (впливовій конференції з питань комп'ютерної безпеки) показали можливість приховання rootkit (до них дійдемо) за допомогою запису до MBR.

І вже в грудні 2007, за твердженням спеціалістів з Verisign iDefense Intelligence Team, з'явився новий різновид шкідливих програм, які використовують давно забутий спосіб автозавантаження. Виявлений ними троянець Mebroot створено з використанням коду, продемонстрованого на Black Hat.

Троянець встановлюється на перший сектор жорсткого диска, до MBR, завантажується при його зчитуванні в пам’ять, після чого вносить зміни в ядро Windows так, щоб ускладнити своє виявлення антивірусним ПЗ.

Багато табличних і графічних редакторів, системи проектування, текстові процесори мають свої макро-мови для автоматизації виконання дій, що повторюються. Ці макро-мови часто мають складну структуру і розвинений набір команд. Макро-віруси є програмами на макро-мовах, вбудованих в такі системи обробки даних. Для свого розмноження віруси цього класу використовують можливості макро-мов і з їх допомогою переносять себе з одного зараженого файлу (документа або таблиці) до інших.

Класифікація файлових вірусів за способом зараження

За способом зараження файлові віруси (віруси, що впроваджують свій код у виконувані файли: командні файли, програми, драйвери, початковий код програм і ін.) розділяють на перезаписувачі, паразитичні, віруси-посилання, віруси-компаньони, а також віруси, що вражають початкові тексти програм і компоненти програмного забезпечення (VCL, LIB і ін.).

— Перезаписуючі віруси. Віруси даного типу записують своє тіло замість коду програми, не змінюючи назви виконуваного файлу, унаслідок чого початкова програма перестає запускатися. При запуску програми виконується код вірусу, а не сама програма.

— Віруси-компаньйони. Компаньон-віруси, як і перезаписуючі віруси, створюють свою копію на місці програми, що заражається, але, на відміну від перезаписувачів, не знищують оригінальний файл, а перейменовують або переміщають його. При запуску програми спочатку виконується код вірусу, а потім управління передається оригінальній програмі. Можливе існування і інших типів вірусів-компаньйонів, що використовують інші оригінальні ідеї або особливості інших операційних систем. Наприклад, PATH-компаньйони, які розміщують свої копії в основному каталозі Windows, використовуючи той факт, що цей каталог є першим в списку PATH, і файли для запуску Windows в першу чергу шукатиме саме в ньому. Даними способом самозапуску користуються також багато комп'ютерних черв'яків і троянські програми.

— Link-віруси. Як і компаньон-віруси, не змінюють код програми, а примушують операційну систему виконати власний код, змінюючи адресу місцеположення на диску зараженої програми, на власну адресу. Після виконання коду вірусу управління зазвичай передається програмі, що викликається користувачем.

— Паразитичні віруси. Паразитичні віруси — це файлові віруси, які змінюють вміст файлу додаючи в нього свій код. При цьому заражена програма зберігає повну або часткову працездатність. Код може упроваджуватися в початок, середину або кінець програми. Код вірусу виконується перед, після або разом з програмою, залежно від місця впровадження вірусу в програму.

— Віруси, що вражають початковий код програм. Віруси даного типу вражають або початковий код програми, або її компоненти (OBJ-, LIB-, DCU- файли) а так само VCL і ActiveX-компоненти. Після компіляції програми виявляються в неї вбудованими. На цей час широкого поширення не набули.

Завантажувальні віруси

Відомі на даний момент завантажувальні віруси заражають завантажувальний (boot) сектор гнучкого диска і boot-сектор або Master Boot Record (MBR) вінчестера. Принцип дії завантажувальних вірусів заснований на алгоритмах запуску операційної системи при включенні або перезавантаженні комп'ютера — після необхідних тестів встановленого устаткування (пам'яті, дисків і т.д.) програма системного завантаження прочитує перший фізичний сектор завантажувального диска (A:, C: або CD-ROM залежно від параметрів, встановлених в BIOS Setup) і передає на нього управління.

Черв’яки

Комп'ютерний черв'як - комп'ютерна програма, здатна до самовідтворення. Вона використовує мережу, щоб відправити свої копії іншим вузлам (комп'ютерним терміналам в мережі) і може робити це без будь-якого втручання користувача. На відміну від вірусу, черв’якам не потрібно приєднуватися до існуючої програми. Черв'яки завжди шкодять мережу (хай навіть лише споживаючи пропускну спроможність), тоді як віруси завжди заражають або зіпсовані файли на атакованому комп'ютері.

Багато створених черв'яків здатні виключно до розповсюдження і не намагаються змінити системи, через які проходять. Проте, як показали черв'як Morpiса і Mydoom, мережний трафік і інші ненавмисні ефекти можуть часто викликати серйозний збій. **Корисне навантаження** — код, призначений для чогось більшого, ніж просте поширення черв'яка – може видаляти файли на зараженій системі (приклад — черв'як ExploreZip), шифрувати файли (криптовірусна атака з метою здирництва), або посилати документи електронною поштою. Дуже популярне корисне навантаження для черв'яків - встановити чорний хід в зараженому комп'ютері, щоб дозволити створення "зомбі" під управлінням автора черв'яка (Sobig і Mydoom). Мережі таких машин часто називають botnets. Їх зазвичай використовують спамери для поширення junk-повідомлень або для маскуваня адреси своїх

веб-вузлів. Саме тому спамерів вважають джерелом фінансування створення таких черв'яків, і автори черв'яків були спіймані при спробі продажу списків IP адрес заражених машин. Інші пробують шантажувати компанії можливими DoS-атаками.

2. Історія вірусів.

Думок з приводу народження першого комп'ютерного вірусу дуже багато. Нам напевно відомо тільки одне: на машині Чарльза Беббіджа, який вважається винахідником першого комп'ютера, вірусів не було, а на Univax 1108 і IBM 360/370 в середині 1970-х років вони вже були. Не дивлячись на це, сама ідея комп'ютерних вірусів з'явилася значно раніше. Відправною крапкою можна вважати праці Джона фон Неймана з вивчення математичних автоматів, що самовідтворюються. Ці праці сталі відомі в 1940-х роках. А в 1951 р. знаменитий учений запропонував метод, який демонстрував можливість створення таких автоматів. Пізніше, в 1959 р., журнал "Scientific American" опублікував статтю Л.С. Пенроуза, яка також була присвячена механічним структурам, що самовідтворювалися. На відміну від раніше відомих робіт, тут була описана проста двовимірна модель подібних структур, здібних до активації, розмноження, мутацій, захоплення. Пізніше, слідами цієї статті інший учений - Ф.Ж. Шталь - на практиці реалізував цю модель за допомогою машинного коду на IBM 650.

Необхідно відзначити, що із самого початку ці дослідження були направлені зовсім не на створення теоретичної основи для майбутнього розвитку комп'ютерних вірусів. Навпаки, учені прагнули удосконалити мир, зробити його більш пристосованим для життя людини. Адже саме ці праці лягли в основу багатьох пізніших робіт з робототехніки й штучного інтелекту. І в тому, що подальші покоління зловжили плодами технічного прогресу, немає провини цих чудових учених.

Початок 70-х років

На початку 1970-х років в прототипі сучасного інтернету - військовій комп'ютерній мережі APRAnet - був виявлений вірус Creeper. Написана для колись популярної операційної системи Tenex, ця програма була в змозіі самостійно ввійти до мережі через модем і передати свою копію віддаленій системі. На заражених системах вірус виявляв себе повідомленням: "IM THE CREEPER : CATCH ME IF YOU CAN". Пізніше для видалення настирливого, але в цілому нешкідливого вірусу, невідомим була створена програма Reaper. За своєю суттю це був вірус, що виконував деякі функції, властиві антивірусу: він розповсюджувався обчислювальною мережею і, у разі виявлення тіла вірусу Creeper, знищував його.

1974 рік. На мейнфреймах цього часу з'являється програма, що отримала назву "кролик" (Rabbit). Це ім'я вона отримала тому, що окрім розмноження і розповсюдження носіями інформації, вона нічого не робила. Правда, швидкість її розмноження цілком виправдовувала назву. Ця програма клонувала себе, займала системні ресурси і таким чином знижувала продуктивність системи. Досягнувши певного рівня розповсюдження на зараженій машині, "кролик" нерідко викликав збій в її роботі.

Початок 80-х років

Комп'ютери стають все більш і більш популярними. З'являється все більше і більше програм, авторами яких є не фірми-виробники програмного забезпечення, а приватні особи. Розвиток телекомунікаційних технологій дає можливість відносно швидко і зручно поширювати ці програми через сервери загального доступу — BBS (Bulletin Board System). Пізніше, напівлюбительські, університетські BBS переростають в глобальні банки даних, що охоплюють практично всі розвинені країни. Вони забезпечують швидкий обмін інформацією навіть між найвіддаленішими точками планети.

1981 рік

Широке розповсюдження комп'ютерів марки Apple II привернуло увагу до цієї платформи авторів вірусів. Не дивно, що перша в історії дійсно масова епідемія комп'ютерного вірусу відбулася саме на Apple II. Вірус Elk Cloner записувався в завантажувальні сектори дискет, до яких йшло звернення. В ті часи це здавалося неймовірним і викликало у рядових користувачів стійкий зв'язок між вірусами і позаземними цивілізаціями, що намагаються завоювати світ. Враження від вірусу посилювалося його проявами: Elk Cloner перевертав зображення на екрані, примушував текст блимати, виводив різноманітні загрозливі повідомлення.

1983 рік

Лен Ейделман вперше використовує термін "вірус" щодо до комп'ютерних програм, що саморозмножуються. 10 листопада 1983 р. Фред Коен, родоначальник сучасної комп'ютерної вірусології, на семінарі з комп'ютерної безпеки в Лехайському університеті (США) демонструє на системі VAX 11/750 вірусоподібну програму, здатну впроваджуватися в інші об'єкти. Роком пізніше, на 7-ій конференції з безпеки інформації, він дає наукове визначення терміну "комп'ютерний вірус" як „програми, здатної "заражати" інші програми за допомогою їх модифікації з метою впровадження своїх копій".

1986 рік

Зареєстрована перша глобальна епідемія вірусу для IBM-сумісних комп'ютерів. Вірус Brain, що заражає завантажувальні сектори дискет, протягом декількох місяців розповсюдився практично по всьому світу. Причина такого "успіху" полягала в повній непідготовленості комп'ютерного суспільства

до зустрічі з таким явищем, як комп'ютерний вірус: антивірусні програми ще не набули такого широкого поширення як зараз, а користувачі, у свою чергу, не дотримувалися основних правил антивірусної безпеки. Ефект від епідемії, що відбулася, посилювався поганим знайомством суспільства і невивченістю феномена "комп'ютерний вірус". Услід за виявленням Brain один за іншим стали з'являтися науково-фантастичні романи, присвячені вірусам.

Вірус Brain був написаний в Пакистані 19-річним програмістом Баситом Фарук Алві (Basit Farooq Alvi) і його братом Амжадом (Amjad), що залишили у вірусі текстове повідомлення, що містить їх імена, адресу і телефонний номер. Як затверджували автори вірусу, що працювали в компанії з продажу програмних продуктів, вони вирішили з'ясувати рівень комп'ютерного піратства у себе в країні. Крім зараження завантажувальних секторів і зміни міток (label) дискет на фразу (c) Brain вірус нічого не робив: він не надавав ніякої побічної дії і не псував інформацію. На жаль, експеримент швидко вийшов з-під контролю і виплеснувся за межі Пакистану.

Цікаво, що вірус Brain був також і першим вірусом-невидимкою. При виявленні спроби читання зараженого сектора диска вірус непомітно "підставляв" його незаражений оригінал.

У 1988 р. були зафіксовані перші випадки розповсюдження т.зв. вірусних містифікацій (Virus Hoax). Це вельми цікавий феномен, заснований на розповсюдженні помилкових чуток про появу нових, надзвичайно небезпечних комп'ютерних вірусів. По суті справи ці чутки і були свого роду вірусами: налякані користувачі поширювали такі повідомлення всім своїм знайомим з надзвичайною швидкістю. Навряд чи варто зупинятися на тому, що містифікації не наносять комп'ютерам ніякого збитку. Проте, разом з тим, вони забивають канали передачі даних, нервують інших користувачів і дискредитують людей, що повірили в ці чутки.

Один з перших жертв такого характеру належить нікому Mike RoChenle (псевдонім схожий на англійське слово "microchannel"), який в жовтні 1988 р. розіслав на станції BBS велику кількість повідомлень про нібито існуючий вірус, який передається від модему до модему і використовує для цього швидкість 2400 бітів на секунду. В якості панацеї пропонувалося якнайскоріше перейти на використання модемів зв швидкістю 1200 бітів на секунду. Як це ні смішно, багато користувачів дійсно послухалися цієї поради.

Іншим жертвом з цієї ж області стало попередження, випущене Робертом Моррісом (Robert Morris) про вірус, що розповсюджується по електричній мережі, змінює конфігурацію портів і напрямок обертання дисководів. Згідно повідомленню, всього за 12 хвилин вірус встиг уразити 300 000 комп'ютерів в штаті Дакота (США).

Листопад 1988. Поголовна епідемія справжнього мережного вірусу, що отримав назву „черв'як Морріса". Вірус заразив більше 6000 комп'ютерних систем в США (включаючи Дослідницький центр NASA) і практично паралізував їх роботу. Унаслідок помилки в коді вірусу він, як і вірус-черв'як "Christmas Tree", необмежено розсилав свої копії по інших комп'ютерах мережі, запускав їх на виконання і таким чином повністю забирав під себе всі мережеві ресурси. Для свого розмноження вірус використовував помилки в системі безпеки операційної системи Unix для платформ VAX і Sun Microsystems. Крім помилок в Unix, вірус використовував і інші оригінальні ідеї, наприклад, підбір паролів користувачів (із списку, що містить 481 варіант) для входу в системи під чужим ім'ям. Загальні збитки від вірусу Морріса були оцінені в 96 мільйонів доларів.

Нарешті, 1988 р. ознаменувався появою однієї з найбільш відомих антивірусних програм - Dr. Solomons Anti-Virus Toolkit. Програма була створена англійським програмістом Аланом Соломоном (Alan Solomon), завоювала величезну популярність і проіснувала до 1998 р., коли компанія була поглинена іншим виробником антивірусів - американською Network Associates (NAI).

У червні 1994 року почалася епідемія OneHalf. OneHalf – написаний під DOS поліморфний комп'ютерний вірус (гібрид бутового і файлового вірусів). Відомий своїм особливим корисним навантаженням, яке кодує певні частини жорсткого диска, однак розшифровує їх, коли з ними працюють, тому користувач нічого не помічає. Кодування здійснюється шляхом порозрядного XORування випадково згенерованим ключем, а розкодування виконується повторним XORуванням з тим самим ключем. Проте, недбала дезинфекція призводитиме до втрати даних, тому що, якщо користувач не бере це кодування до уваги, то він знищує тільки вірус, який служить також для розшифровки і доступу до даних. Після кодування однієї половини HDD за особливих умов вірус пише таке повідомлення: Dis is OneHalf. Press any key to continue ...

Січень 2003 — Slammer:

(З новин): „У минулі вихідні була здійснена одна з найкрупніших вірусних атак на Інтернет за всю його історію. В результаті активізації черв'яка Slammer швидкість роботи Мережі значно сповільнилася, а деякі регіони, наприклад, Південна Корея, виявилися практично відрізаними від інтернету.

Вірусна атака почалася в 0:30 за часом Східного побережжя США або в 8:30 за московським часом. Де знаходилося джерело зараження, до цих пір точно не відомо. Деякі фахівці з комп'ютерної

безпеки припускають, що вірус розповсюджувався з території США, інші ж вважають, що його батьківщина знаходиться десь в Азії.

За лічені хвилини черв'як, що використовує уразливість в СУБД Microsoft SQL Server 2000, заповнив інтернет. Незважаючи на малий розмір вірусу - всього 376 байт, він зміг створити в каналах передачі даних справжні пробки, оскільки після зараження комп'ютера він починає розсилати свій код по випадкових IP-адресах в нескінченному циклі. Якщо по якій-небудь з адрес виявлявся вразливий комп'ютер, він заражався і теж починав розсилати копії вірусу.

Все це привело до великомасштабного зростання трафіку. На піку активності черв'яка на один сервер могли приходити сотні запитів в хвилину. Не витримавши збільшеного навантаження, деякі сервери просто падали. В цей час тільки в США втрачалося до 20% IP-пакетів, що вдесятеро перевищує нормальний рівень. За наявними даними, від атаки постраждали і п'ять з тринадцяти кореневих DNS-серверів.

Найсильніше від атаки Slammer постраждала Південна Корея, де інтернетом користуються сім з кожних десяти жителів. Проте в суботу ця країна виявилася практично відключеною від глобальної мережі. Зокрема, під натиском Slammer обрушилася мережа найбільшого південнокорейського провайдера KT Corp. Серйозно постраждали також провайдери Hanaro Telecom Inc. і Thrunet, що займають, відповідно, друге і третє місця на південнокорейському ринку мережевих послуг. Працездатність була відновлена вже до суботнього вечора, проте швидкість роботи інтернету ще довго залишалася низькою.

Відключення інтернету в Південній Кореї позначилося і на роботі Мережі у всьому азіатському регіоні - швидкість роботи інтернету тут була найменшою. У Європі сповільнення також було серйозним, проте до південнокорейських масштабів катастрофа не дійшла. Американські провайдери інтернету, в більшості своїй, стабільно працювали під час атаки, хоча швидкість передачі даних була значно нижча за звичайну.

Протягом суботи атака Slammer поступово зійшла нанівець, проте експерти по комп'ютерній безпеці побоюються, що вірус ще повернеться. Суботня атака нанесла значно менше збитку, ніж якби вона проводилася в розпал робочого тижня. Цілком імовірно, що суботня епідемія - всього лише репетиція перед справжньою атакою на інфраструктуру Мережі, від якої залежить бізнес безлічі компаній по всьому світу.

Однією з причин катастрофічних наслідків атаки Slammer є неувага системних адміністраторів до регулярного оновлення програмного забезпечення. Про дірку в MS SQL Server 2000 стало відомо ще минулого літа, а латочка до неї міститься у випущеному Microsoft пакеті оновлень Service Pack 3. Проте, згідно з відомою приказкою про мужика і грім, встановленням патчів адміністратори зайнялися лише після атаки Slammer. Проте вдалося це небагатьом: сайт Microsoft, звідки тільки і можна було узяти патч, виявився перевантаженим." (http://security.compulenta.ru/37085/)

Термін MalWare 2.0 (Лабораторія Касперського) описує сучасну модель функціонування комплексів шкідливих програм, що сформувалися наприкінці 2006 р. Першими та найяскравішими представниками MalWare 2.0 стали черви Bagle, Warezov и Zhelatin.

Основні характеристики цієї моделі:

- відсутність єдиного центру керування мережею заражених комп'ютерів;
- активна протидія спробам вивчити шкідливий код та перехопити керування ботнетом;
- короткочасні масові розсилки шкідливого коду;
- грамотне використання засобів соціальної інженерії;
- використання різних способів розповсюдження шкідливих програм та поступова відмова віт найпомітніших з них (наприклад, від електронної пошти);
- використання різних (а не одного універсального) модулей для виконання різних шкідливих функцій.

Розвиток MalWare 2.0 породжує низку проблем для антивірусної індустрії. Найважливіша з них — нездатність традиційних антивірусних рішень, побудованих виключно на сигнатурному чи евристичному методах аналізу файлів, надійно протидіяти атакам MalWare 2.0, не кажучи вже про лікування уражених систем.

Тенденції 2008 року (за даними ЛК):

- лавиноподібне зростання кількості шкідливих програм типу Trojan-SMS;
- значне поширення шкідливих програм, націлених на викрадення паролів до он-лайн-ігор;
- різке зростання кількості черв'яків призвело до того, що вони стали становити більшість в класі VirWare. Net-Worm'и все частіше стали використовувати для розповсюдження зламані сайти і соціальні мережі;

Тенденції 2009 року:

— епідемія трояну Trojan-Downloader.JS.Gumblar.x;

— епідемія черв'яка Net-Worm.Win32.Kido.ih (Conficker);

— лідер атак: DoS.Generic.SYNFlood (71.192% усіх атак, відбитих IDS (intrusion detection system) програмного комплексу KIS 2010); третю позицію утримує Intrusion.Win.MSSQL.worm.Helkern (він же — вже згадуваний Slammer);

Лекція 2.

1. Типові пошкодження ОС, програм та даних вірусами
2. Способи зараження файлів та комп'ютерних систем.
3. Способи маскування вірусів.

1. Типові пошкодження ОС, програм та даних вірусами

1. „Ігрові" ефекти

Історично перший різновид шкоди, яку наносили віруси комп'ютерам. Перші віруси були радше жартами і писалися з метою продемонструвати колегам чи приятелям свої програмістські навички. І мало що може зрівнятися у переконливості, ніж раптова зміна поведінки комп'ютера і різні жартівливі ефекти, такі як виведення на екран різних цікавих повідомлень, самовільне блимання тексту тощо. Такі віруси не чіпали даних чи програм, і досить легко видалялися. Тим не менше, зусилля, які слід було для цього прикласти, збивали користувачів з робочого настрою, змушували нервувати. Тому навіть ці віруси — шкідливі (приклади — Creeper, Elk Cloner).

Іноді „ігрові" ефекти використовуються в якості побічних у більш серйозних вірусах. Наприклад, вірус OneHalf також передбачав виведення на екран повідомлення "Dis is OneHalf. Press any key to continue", але робив це вкрай рідко (після зашифровки половини диску на кожній четвертій зараженій машині, якщо число і значення таймера кратні чотирьом). Подібний прийом використовував черв'як Christmas Tree, що виводив на екран зараженого комп'ютера зображення різдвяної ялинки.

2. Руйнування даних та структур даних у файлі, файлової системи тощо

Файл може бути необоротно зіпсований в результаті помилки вірусу при копіюванні свого коду в середину файлу. Особливо це актуально для cavity-вірусів, як можуть неточно розрахувати довжину порожнини і зачепити частину коду самого файлу.

Про OneHalf: При всій ретельності прописування процедур зараження і шифровки-розшифровки диска при практичних зараженнях дисків виявлено, як мінімум, дві помилки. Одна з них полягає в тому, що деякі диски об'ємом 420 мегабайт заражалися некоректно (при зараженні невірно визначалася кількість секторів MBR і останній сектор тіла вірусу «відбивався» на сектор, зайнятий Partition Table, що приводило диск в неробочий стан). Слід врахувати, що в цьому випадку вірус встигав коректно перенести оригінальний MBR в один з прихованих секторів, і можна було відновити працездатність диска за допомогою перенесення MBR на своє місце вручну програмою Diskedit. Другою помилкою було те, що при шифруванні деяких дисків вірус «помилявся» і шифрував останні 2 доріжки з ключем, відмінним від записаного в MBR. При наступному запуску і роботі із зашифрованими доріжками він коректно виправляв цю помилку, проте, якщо цей момент співпадав з розшифровкою доріжок антивірусами, то останні 2 доріжки можна було вважати втраченими.

Взагалі, з вірусом OneHalf пов'язана цікава історія. В принципі, сам по собі цей вірус не дуже шкідливий, оскільки сам розшифровує зашифровані файли, після чого вони працюють абсолютно коректно. Звісно, його, як і будь-яку програму, може проглючити, і тоді наслідки можуть бути дуже серйозними (див. вище), і, звісно, слід намагатися здихатися його якнайшвидше. Тут і виникли проблеми.

Справа в тому, що найбільше від пов'язаних із діяльністю вірусу OneHalf втрат інформації постраждали користувачі антивірусу Dr. Web версій включно до 1.6 — єдиного

антивіруса-поліфага, що вбивав OneHalf. Це пояснювалося тим, що в цих версіях автор антивірусу Ігор Данилов не передбачив дій по відновленню зашифрованої частини диску, а просто видаляв вірус, після чого дані виявлялися втраченими, оскільки нема кому було їх розшифровувати (ключ знищувався разом із вірусом). Звісно, в цих втратах інформації звинуватили сам вірус, а в наступних версіях Dr. Web'а цей недолік було усунуто.

Дуже небезпечним є вірус Virus_Win32_Gpcode_ak, який шифрує вміст файлів (перелік розширень налічує більше 80 пунктів) та створює текстовий файл, в якому пропонується адреса, за якою треба перерахувати певну не дуже велику суму (від 500 до 2000 рублів) грошей за програму-декодер. Спершу (2004 р.) зловмисник використовував власний алгоритм шифрування, розкрити який спеціалістам вдалося без зусиль, але потім перейшов до використання значно більш потужного алгоритму за алгоритмом RSA (відкритий ключ – закритий ключ). Перші версії (з довжиною ключа 56, 67 і аж до 640 бітів) спеціалістам Лабораторії Касперського вдалося факторизувати (себто навчитися знаходити за наявним відкритим ключем закритий), але в останніх зловмисник дійшов до довжини ключа 660 та 1024 біти, які розшифрувати дуже і дуже складно, а на повний перебір потрібні роки безперервної роботи всіх комп'ютерів світу (для довідки – знаходження ключа довжиною 640 бітів методом перебору потребує 30 років на комп'ютері з процесором на 2,2 ГГц). **Задача «зламу» ключа RSA-1024, дуже складна та фундаментальна криптографічна проблема.** За оцінками ЛК, для зламу такого ключа потрібен майже рік роботи п'ятнадцяти мільйонів сучасних комп'ютерів.

Поширення:

Спам зі шкідливою програмою розсилався по електронним адресам, зібраним на сайті job.ru. Люди, зацікавлені в новій роботі, отримували відповідні своїм анкетам пропозиції начебто від представників поважних західних компаній. До них додавася вкладений файл з анкетою, яку треба було заповнити, щоб вирішити питання розміру зарплатні. Втриматися від потенційно небезпечної дії за такої подачі дуже важко.

Насправді доданий файл anketa.doc являв собою троянску програму Trojan-Dropper.MSWord.Tored.a. Після його відкриття спрацьовував шкідливий макрос, що встановлював іншого троянця — Trojan-Downloader.Win32.Small.crb. Він завантажував в систему Gpcode з певної, постійно змінюваної адреси.

Жодних видимих користувачу змін після відкриття аттачменту не відбувалося, завантаження троянців і шифрування відбувалося через певний час після відкриття файлу, тому в переважній більшості випадків користувачі не запідозрювали, що шифрування файлів та лист-відповідь на анкету на job.ru пов'язані. Тому механізм зараження комп'ютерів довго не вдавалося зрозуміти.

Втрати даних, програми-відновлювачі стертих файлів. Проект «Stop Gpcode», утиліти PhotoRec (відновлення стертих файлів) і StopGpcode (їх розташування по звичних місцях).

3. Руйнування програм, уповільнення їх роботи, приведення програм у неробочий стан

Яскравим прикладом таких вірусів є віруси-перезаписувачі, що записують свій код замість коду файлу, що заражається, знищуючи його вміст. Природно, що при цьому файл перестає працювати і не відновлюється. Хоча, такі віруси дуже швидко виявляють себе, оскільки операційна система і застосування досить швидко перестають працювати, вони дуже небезпечні, оскільки через них доводиться переустановлювати знищену програму або ОС.

Іншим способом серйозно нашкодити користувачам є сповільнення роботи конкретної програми чи ОС в цілому за рахунок дописування у код програми, наприклад, довгих циклів. З метою сповільнення ОС вірус запускає процес, який може або взагалі бути прихованим від користувача, або маскуватися під інший процес, найкраще системний, на який користувач не одразу зверне увагу. Інколи обходяться без цього, спеціально обираючи

специфічні імена, на кшталт THIS_IS_VIRUS. В останньому випадку є сенс казати про змішаний сповільнююче-„ігровий” ефект.

Якщо руйнування програми — це заміщення коду програми вірусом і виконання вірусу замість програми, то приведення програми у неробочий стан — це блокування вірусом запуску цієї програми, наприклад, через заміщення шляху до файлу програми на шлях до свого файлу, або просто в нікуди, що може призвести до збоїв у роботі великих багатомодульних програм. Також досить ефективно прописувати себе замість драйверів пристроїв, після чого вони перестають працювати, а користувач звинувачує у всьому „залізо”.

Збій в роботі програми також можна викликати, використовуючи техніку „переповнення стека/буфера” (stack/buffer overflow). В комп’ютерній безпеці та програмуванні, **переповненням буфера** називається програмна помилка, яка може призвести до виняткової ситуації доступу до пам’яті та завершення програми або, якщо користувач зробив це навмисно, до можливого порушення безпеки.

Переповнення буфера — це аномальна ситуація, коли процес намагається записати свої дані за межі буфера фіксованої довжини. В результаті надлишкові дані записуються поверх сусідніх даних. Заміщені таким чином дані можуть бути іншими буферами, змінними, даними про роботу програми, що може призвести до аварійного завершення програми або одержання невірних результатів. Також переповнення може бути викликане вхідними даними, спеціально розробленими для виконання шкідливого коду чи щоб змусити програму поводитися непередбачувано.

Переповнення буфера спричиняють вразливість програмних продуктів і є базою багатьох експлойтів. Втім, ретельна перевірка розміру буфера програмістом, компілятором чи середовищем виконання може запобігти цьому. Також у Windows Vista вбудовано механізм імунізації пам’яті ASLR (Address Space Layout Randomization). Під час запуску система поміщає виконувані файли та DLL в випадкові ділянки пам’яті, захищаючись таким чином від нападу на наперед відомі адреси.

4. Переповнення пам’яті, засмічення каналів зв’язку

Поголовна епідемія справжнього мережного вірусу, що отримав назву „черв'як Морріса”. Вірус заразив більше 6000 комп'ютерних систем в США (включаючи Дослідницький центр NASA) і практично паралізував їх роботу. Унаслідок помилки в коді вірусу він, як і вірус-черв'як "Christmas Tree", необмежено розсилав свої копії по інших комп'ютерах мережі, запускав їх на виконання і таким чином повністю забирав під себе всі мережеві ресурси. Для свого розмноження вірус використовував помилки в системі безпеки операційної системи Unix для платформ VAX і Sun Microsystems. Крім помилок в Unix, вірус використовував і інші оригінальні ідеї, наприклад, підбір паролів користувачів (із списку, що містить 481 варіант) для входу в системи під чужим ім'ям. Загальні збитки від вірусу Морріса були оцінені в 96 мільйонів доларів.

5. Порушення функціонування ОС

Два останніх байта boot- та MBR-секторів містять код 55AAh. Якщо його затерти, система перестане з цього диска завантажуватися (див. далі). Це інколи використовується авторами вірусів для того, щоб комп’ютер не міг завантажитися з жорсткого диска. Методика ефективна у поєднанні з „латентним періодом” — певний час вірус лише розмножується, а потім всі заражені комп’ютери, як по команді, перестають завантажуватися.

У випадку Windows-систем шкідливі програми атакують системний реєстр. Інколи вони обмежуються простим його засміченням, що вже спричиняє затримки в завантаженні та роботі системи, інколи — заміщають собою важливі системні записи.

6. Фізичне пошкодження комп'ютерів

Яскравим прикладом таких вірусів є CIH („Чорнобиль"). Він поширюється через файли формату Portable Executable (EXE) під Windows 95, Windows 98, и Windows ME. CIH не поширюється під Windows NT, Windows 2000, Windows XP чи Windows Vista.

CIH заражає виконавчі файли, розділяючи свій код на маленькі фрагменти, що вставляються в проміжки між розділами, зазвичай присутні у PE файлах, та записуючи невелику процедуру ре-асемблювання та таблицю розташування сегментів свого коду у невикористовуване місце в хвості PE-заголовка. Через це CIH прозвали "Spacefiller". Розмір вірусу — приблизно 1 кілобайт, але через його просунутий метод мульти-карієсного зараження заражені файли зовсім не збільшуються. Вірус також використовує метод перескакування з процесорного кільця 3 (рівень прикладних програм) на кільце 0 (рівень ядра ОС) для перехоплення системних викликів.

Корисне навантаження цього дуже небезпечного вірусу спершу заповнює перший мегабайт (1024KB) жорсткого диску, починаючи з сектора 0, нулями. Це призводить до видалення інформації про таблицю логічного розбиття диску і завішує машину..

Друга частина навантаження намагається перезаписати Flash BIOS. Через, можливо, ненавмисну рису цього вірусу, в BIOSax, які він може вдало перезаписати, критичний код часу завантаження заміщується на сміття. Це спрацьовує лише на деяких машинах. Багато казали про особливу вразливість комп'ютерів з материнськими платами на базі чіпсета Intel 430TX, але найважливішим чинником успішного доступу вірусу до флеш-пам'яті є тип Flash ROM чіпа. Різні Flash ROM чіпи (або сімейства чіпів) мають різні процедури дозволу запису. CIH не робить жодних спроб визначити тип Flash ROM чіпу и має в своєму розпорядженні лише один варіант процедури запису.

Якщо спрацьовує перша частина навантаження, перезаписана інформація губиться повністю. Якщо перший логічний диск відформатований під FAT32 і має обсяг більше 1ГБ, перезаписується лише MBR, таблиця розбиття, бут-сектор першого логічного диска та перша копія FAT першого диска. MBR та бут-сектор можна просто замінити стандартними версіями, таблицю розбиття — перебудувати шляхом перегляду всього вінчестера, а першу копію FAT — відновити з другої копії. Тобто, повне відновлення всіх даних без втрат може бути зроблено автоматично за допомогою утиліт типу Fix CIH. Якщо ж перший диск не FAT32 або менший за 1GB, дані користувача на цьому диску вціліють, але без кореневого каталогу та FAT її буде складно відновити в повному обсязі, особливо в разі великої фрагментації даних.

Якщо спрацьовує друга частина корисного навантаження, комп'ютер взагалі не запуститься. Треба або викликати спеціаліста для перепрограмування чіпа, або замінити його, оскільки системи, які вміє вражати вірус, ще не мають засобів відновлення BIOS'а.

7. Загроза конфіденційності користувачів

Ця проблема стосується вже радше не вірусів, які їх творці просто випускають в світ за принципом „вистрілити і забути", а шпигунських і троянських програм, що приносять своїм авторам дивіденди у вигляді відкритих до зовнішнього керування комп'ютерів чи паролей до електронної пошти або банківських рахунків. Але, оскільки сучасна комп'ютерна вірусологія боротьбою з самими лише класичними вірусами не вичерпується, видається логічним розглянути, в принципі, найбільшу небезпеку, яку несуть шкідливі програми, а саме загрозу крадіжки персональних даних користувачів з подальшим їх використанням для нанесення їм моральної або матеріальної шкоди. Ця загроза може бути реалізована кількома шляхами.

— соціальна інженерія;

— засилання на комп'ютер шпигунських програм (кейлогерів, сніферів, бекдорів тощо);

— „викрадення” веб-браузера (при вході на веб-сторінку, що містить спеціальні скрипти або завантажує особливим чином сконструйовані HTTP-“cookies” чи LSO (Local Shared Object, або, неформально, „Flash-cookies”)).

8. Інше

Вірус Virut.a

Вірус заражає виконувані файли Windows. Після зараження намагається з’єднатися з певним IRC-сервером. Якщо йому це вдається, він приймає команди від зловмисника і виконує їх. Вміє завантажувати на комп’ютер користувача шкідливий код та відкривати з комп’ютера користувача вказані зловмисником URL-адреси

Worm.Win32.Fujack.a

Черв’як заражає файли з розширеннями «.exe», «.scr», «.pif», «.com» на всіх фіксованих дисках — за винятком файлів в деяких папках.

При зараженні файлів черв’як записує свій виконуваний файл перед їх оригінальним змістом. Також вірус намагається заразити файли в мережних папках. Для цього він намагається підключитися до віддалених комп’ютерів в мережному оточенні, використовуючи такі імена як Administrator, Guest, Admin, Root та великий набір стандартних паролів. Проникає на з’ємні диски, створюючи setup.exe та autorun.inf.

2. Способи зараження файлів та комп’ютерних систем

Файлові віруси

За способом зараження файлів віруси поділяються на:
перезаписувачі (overwriting);
паразити (parasitic);
віруси-компаньони (companion);
віруси-посилання (link);
віруси, що заражають об’єктні модулі (OBJ);*
віруси, що заражають бібліотеки компіляторів (LIB);*
віруси, що заражають вихідні тексти програм.*
* — екзотика.

Тому в своїй массі ФВ — віруси, що заражають файли трьох типів: командні файли (BAT), завантажувальні драйвери (SYS) та виконавчі двійкові файли (COM, EXE).

— Overwriting

Даний метод зараження є найбільш простим: вірус записує свій код замість коду файлу, що заражається, знищуючи його вміст. Природно, що при цьому файл перестає працювати і не відновлюється. Такі віруси дуже швидко виявляють себе, оскільки операційна система і застосування досить швидко перестають працювати.

— Parasitic

До паразитичних відносяться всі файлові віруси, які при розповсюдженні своїх копій обов'язково змінюють вміст файлів, залишаючи самі файли при цьому повністю або частково працездатними. Основними типами таких вірусів є віруси, що записуються в початок файлів (prepending), в кінець файлів (appending) і в середину файлів (inserting). У свою чергу, впровадження вірусів в середину файлів відбувається різними методами — шляхом перенесення частини файлу в його кінець або копіювання свого коду в свідомо невживані дані файлу (cavity-віруси).

Впровадження вірусу в початок файлу

Відомо два способи впровадження паразитичного файлового вірусу в початок файлу. Перший спосіб полягає в тому, що вірус переписує початок файлу, що заражається, в його кінець, а сам копіюється в місце, що звільнилося. При зараженні файлу другим способом вірус створює у оперативній пам’яті свою копію, дописує до неї заражуваній файл і зберігає одержану конкатенацію на диск.

Таким чином, при запуску зараженого файлу першим управління отримує код вірусу. При цьому віруси, щоб зберегти працездатність програми, або лікують заражений файл, повторно запускають його, чекають закінчення його роботи і знову записуються в його початок (іноді для цього використовується тимчасовий файл, в який записується знешкоджуваний файл), або відновлюють код програми в пам'яті комп'ютера і настроюють необхідні адреси в її тілі (тобто дублюють роботу ОС).

Впровадження вірусу в середину файлу

Існує декілька методів впровадження вірусу в середину файлу. У найпростішому з них вірус переносить частину файлу в його кінець або «розсовує» файл і записує свій код в простір, що звільнився. Деякі віруси при цьому компресують переносимий блок файлу так, що довжина файлу при зараженні не змінюється.

Другим є „карієсний” метод («cavity»), при якому вірус записується в свідомо невживані області файлу. Вірус може бути скопійований в незадіяні області заголовка EXE-файлу, в «дірки» між секціями EXE-файлів або в область текстових повідомлень популярних компіляторів. Існують віруси, що заражають тільки ті файли, які містять блоки, заповнені яким-небудь постійним байтом, при цьому вірус записує свій код замість такого блоку.

Крім того, копіювання вірусу в середину файлу може відбутися в результаті помилки вірусу, в цьому випадку файл може бути необоротно зіпсований.

Впровадження вірусу в кінець файлу

Найпоширенішим способом впровадження вірусу у файл є дописування вірусу в його кінець. При цьому вірус змінює початок файлу таким чином, що першими виконуваними командами програми, що міститься у файлі, є команди вірусу. У COM-файлі в більшості випадків це досягається шляхом заміни перших трьох (інколи більше) байтів на команду переходу на код вірусу. В EXE-файлі модифікується його заголовок — змінюються значення стартової адреси (точки входу у програму) і довжині файлу (додавання довжині самого вірусу).

Зараження .BAT-файлів:

Віруси дописують в .BAT файл свій набір інструкцій. Особливо небезпечні ті, які працюють з AUTOEXEC.BAT, оскільки це безпосередньо відбивається на роботі системі. Втім, такі віруси досить просто виявити, оскільки BAT-файли без проблем читаються. Але для цього користувач має буті достатньо кваліфікований та обізнаний із тім, що має буті записане в даному BAT-файлі, а що — сторонній код.

Зараження .SYS-файлів

Вірус приписує свій код до тіла файлу та модифікує його заголовок таким чином, що DOS розглядає інфікований файл як ланцюжок з двох або більше драйверів. Такий вірус може буті дуже небезпечнім і живучим, оскільки завантажується в ОП при завантаженні ОС раніше за будь-який антивірус (звісно, якщо він також не є драйвером).

Зараження .COM-файлів

Вірус записує свій код у файл. Відповідно, модифікується початок файлу — замість перших трьох байтів записується трибайтна команда ближнього переходу JMP (передача керування на тіло вірусу), а три оригінальні байті зберігаються в області даних вірусу. Специфіка роботи з COM-файлами полягає в тому, що в DOS їх розмір не може

перевищувати 64 кілобайт. Це пов’язано з тім, що COM-формат передбачає розміщення програмних кодів, даних та стеку в одному сегменті оперативної пам’яті, а розмір сегменту обмежений 64 кілобайтами (від 0x0100 до 0xFFFF). При цьому 256 байтів за адресами від 0x0000 до 0x00FF відводяться під спеціальну службову структуру — Program Segment Prefix (PSP), що містить інформацію, необхідну для коректної роботи програмі (дані туди вносить ОС). Таким чином, якщо розмір COM-файлу + розмір вірусу > 64 Кбайт, то вірусу не вдасться його заразити.

Якщо каталог містить як файл COM, так і файл EXE з таким же ім'ям (не враховуючи розширення), перевага надається файлу COM. Наприклад, якщо каталог містить два файли, названі foo.com і foo.exe, наступна інструкція запустила б foo.com:

C:\>foo.

Якщо користувач бажає виконати foo.exe, він можуть явно вказати повне ім'я файлу:

C:\>foo.exe.

З огляду на таку поведінку за замовчуванням, вірусописці та інші нечисті на руку програмісти іноді для своїх творінь використовують імена, подібні до notepad.com. Вони розраховують на те, що якщо цей файл потрапить до одного каталогу із відповідним EXE-файлом, невичерпно прописані команда запуску або командний файл запустять їх програму, а не текстовий редактор notepad.exe.

Останнім часом деякі вірусописці намагаються скористатися вірогідною відсутністю у сучасних користувачів досвіду роботи із командними файлами COM у поєднанні із знайомством з ім’ям Інтернет-домену .com. Вони надсилають електронні листи з вкладеним файлом, що називається, наприклад, "www.example.com". Необережний користувач Microsoft Windows, який клацає на вкладений файл, не потрапляє на веб-вузол http://www.example.com/, а запускає майстерно написаний та, ймовірно, шкідливий, двійковий командний файл www.example, надаючи йому повний доступ до своєї машини.

Зараження .EXE-файлів

Кілька слів про формат .EXE-файлів. EXE-файли складаються із заголовка, таблиці налаштування та власне програмних кодів і даних. Таблиця налаштування використовується MS-DOS під час роботи з файлом для виконання команд віддаленого переходу чи виклику процедури, для яких потрібно знати не лише зміщення відносно початку сегменту пам’яті, але й адресу сегмента пам’яті (EXE-файли не обмежені одним сегментом). Найбільше нас цікавить саме заголовок. Його структура така:

Байти 0, 1 — містять код 4D5Ah, або MZ — ініціали Марка Збіковськи, керівника колективу розробників однієї з ранніх версій ДОС.

2, 3 — залишок від ділення розміру завантажувального модулю на 512.

4, 5 — розмір файлу в 512-байтних сторінках, округлений вгору.

6, 7 — кількість елементів таблиці налаштування адрес.

8, 9 — розмір заголовка в параграфах (параграф — 16 байтів).

0A, 0B — мінімальна кількість додаткових параграфів, потрібних програмі.

0C, 0D — максимальна кількість додаткових параграфів.

0E, 0F — зміщення сегмента стеку в завантажувальному модулі в параграфах (позначимо SS0).

10, 11 — значення регістру SP (указник стека), яке встановлюється перед передачею керування програмі (SP0).

12, 13 — контрольна сума EXE-файла.

14, 15 — значення регістра IP (лічильник програм) перед передачею керування програмі (IP0).

16, 17 — зміщення сегмента коду в модулі завантаження (CS0).

18, 19 — відстань в байтах від початку файлу до першого елемента таблиці налаштування адрес.

1A, 1B — значення 0, якщо дана частина програми резидентна, чи відмінне від 0 — якщо оверлейна.

Для того, щоб при запуску зараженої програми вірус одержав керування, вихідні значення CS0 та IP0 замінюються точкою входу в код вірусу, а значення SS0 та SP0 перемикаються на власний стек вірусу. Крім того, оскільки довжина завантажувального модуля та довжина файлу також змінюються, слід відкоригувати поля заголовку по зміщенню 02h, 03h, 04h, 05h. Вихідні параметри заголовку зберігаються в області даних вірусу.

— Віруси без точки входу

Окремо слід зазначити досить незначну групу вірусів, що не мають «точки входу» (EPO-віруси — Entry Point Obscuring viruses). До них відносяться віруси, що не змінюють адресу точки старту в заголовку EXE-файлів. Такі віруси записують команду переходу на свій код в яке-небудь місце в середину файлу і отримують управління не безпосередньо при запуску зараженого файлу, а при виклику процедури, що містить код передачі управління на тіло вірусу. Причому виконуватися ця процедура може украй рідко (наприклад, при виведенні повідомлення про яку-небудь специфічну помилку). В результаті вірус може довгі роки «спати» усередині файлу і вискочити на свободу тільки за деяких обмежених умов. Перед тим, як записати в середину файлу команду переходу на свій код, вірусу необхідно вибрати «правильну» адресу у файлі — інакше заражений файл може виявитися зіпсованим.

Відомо декілька способів, за допомогою яких віруси визначають такі адреси усередині файлів, наприклад, пошук у файлі послідовності стандартного коду заголовків процедур мов програмування (C/Pascal), дизасемблювання коду файлу або заміна адрес функцій, що імпортуються.

— Companion

До категорії «companion» відносяться віруси, які не змінюють файлів, що заражаються. Алгоритм роботи цих вірусів полягає в тому, що для файлу, що заражається, створюється файл-двійник, причому при запуску зараженого файлу управління отримує саме цей двійник, тобто вірус. До вірусів даного типу відносяться ті з них, які при зараженні присвоюють файлу яке-небудь інше ім'я, запам'ятовують його (для подальшого запуску файлу-господаря) і записують свій код на диск під ім'ям файлу, що заражається. Наприклад, файл NOTEPAD.EXE перейменовується в NOTEPAD.EXD, а вірус записується під ім'ям NOTEPAD.EXE. При запуску управління отримує код вірусу, який потім запускає оригінальний NOTEPAD.

Можливе існування і інших типів вірусів-компаньйонів, що використовують інші оригінальні ідеї або особливості інших операційних систем. Наприклад, PATH-компаньйони, які розміщують свої копії в основному каталозі Windows, використовуючи той факт, що цей каталог є першим в списку PATH, і файли для запуску Windows в першу чергу шукатиме саме в ньому. Даними способом самозапуску користуються також багато комп'ютерних черв'яків і троянські програми.

— Інші способи зараження

Існують віруси, які жодним чином не пов'язують свою присутність з яким-небудь виконуваним файлом. При розмноженні вони всього лише копіюють свій код в які-небудь каталоги дисків в надії, що ці нові копії будуть коли-небудь запущені користувачем. Іноді ці віруси дають своїм копіям «спеціальні» імена, щоб підштовхнути користувача на запуск своєї копії — наприклад, INSTALL.EXE або WINSTART.BAT. Деякі віруси записують свої копії в архіви (ARJ, ZIP, RAR). Інші записують команду запуску зараженого файлу в BAT-файли. Link-віруси також не змінюють фізичного вмісту файлів, проте при запуску зараженого файлу «примушують» ОС виконати свій код. Цієї мети вони досягають модифікацією необхідних полів файлової системи. Коли заражений файл виконується, вірус

йде постійно завантаженим і записує (зазвичай приховано) файл до диска: цей файл містить вірусний код. Згодом, вірус змінює FAT до посилання перетину інші файли до сектора диска, що містить вірусний код. Результат є, що кожного разу заражений файл виконується, система переходить спочатку до вірусного коду і виконує це. Перехресне з'єднання можна виявити програмою CHKDSK, хоча вірус може використовувати хитрість, щоб приховати зміни, спричинені його наявністю в пам'яті (іншими словами, поки користувач не завантажиться з чистого системного диска, вірус виявлений не буде).

«Лаборатория Касперского» 16.07.08 повідомила про виявлення шкідливої програми, що заражає аудіофайли. Метою зараження є завантаження троянської програми.

Черв'як, що отримав назву Worm.Win32.GetCodec.a, конвертує mp3-файли у формат WMA (при цьому зберігаючи розширення mp3) и додає в них маркер із посиланням на заражену web-сторінку. Активація маркера здійснюється автоматично під час прослуховування файлу і призводить до запуску браузера Internet Explorer, який переходить на інфіковану сторінку, де користувачу пропонується скачати і встановити «кодек». Якщо користувач погоджується, на його комп'ютер завантажується троянська програма Trojan-Proxy.Win32.Agent.arp.

До цього формат WMA використовувався троянськими програмами лише в якості маскування, тобто заражений об'єкт не був музичним файлом. Даний черв'як заражає чисті аудіофайли, що є першим таким випадком і підвищує вірогідність успішної атаки.

Завантажувальні віруси

Завантажувальні віруси заражають завантажувальний (boot) сектор гнучкого диска і boot-сектор або Master Boot Record (MBR) вінчестера. Принцип дії завантажувальних вірусів заснований на алгоритмах запуску операційної системи при включенні або перезавантаженні комп'ютера — після необхідних тестів встановленого устаткування (пам'яті, дисків і т.д.) програма системного завантаження прочитує перший фізичний сектор завантажувального диска (A:, C: або CD-ROM залежно від параметрів, встановлених в BIOS Setup) і передає на нього управління. Загальний принцип — якщо робиться спроба завантажитися із зараженої дискети, вірус заражає MBR жорсткого диска. Потім при старті комп'ютера з вінчестера вірус завантажуватиметься в оперативну пам'ять і заражатиме всі дискети, що проходять через цей комп'ютер, таким чином розмножуючись і поширюючись.

При зараженні дисків завантажувальні віруси «підставляють» свій код замість якої-небудь програми, що одержує керування при завантаженні системи. Принцип зараження, таким чином, однаковий у всіх описаних вище способах: при пере запуску системи вірус «примушує» її записати в пам'ять і віддати управління не оригінальному коду завантажувача, а коду вірусу. Для цього вірус виконує такі дії:

1) під час початкового завантаження системи він записує себе в пам'ять за адресою 0000:7C00h (саме за цією адресою система записує перший сектор нульової доріжки нульової сторони дискети або першого жорсткого диску).

2) Після цього перевстановлює значення регістрів SS та SP на свій власний стек.

3) Відрізає від системи кілька кілобайтів пам'яті (в старших адресах, тобто в кінці) та записує в цю область свій код.

4) Передає керування наступній секції свого коду, вже записаній в основну пам'ять.

5) Ця секція, в свою чергу, спершу перевизначає вектор переривання INT 13h на код вірусу.

6) Записує в пам'ять за адресою 0000:7C00h справжній завантажувальний сектор.

7) Перевіряє, чи заражений вінчестер. Якщо ні — заразити. Якщо так — іти далі.

8) Передає керування справжньому завантажувальному сектору.

Зараження дискет проводиться єдиним відомим способом — вірус записує свій код замість оригінального коду boot-сектора дискети. Вінчестер заражається трьома можливими

способами — вірус записується або замість коду MBR, або замість коду boot-сектора завантажувального диска (зазвичай диска C:), або модифікує адресу активного boot-сектора в таблиці розділів диска (Disk Partition Table), розташованій в MBR вінчестера.

При інфікуванні диска вірус в більшості випадків переносить оригінальний boot-сектор (або MBR) в якій-небудь інший сектор диска (наприклад, в перший вільний). Якщо довжина вірусу більше довжини сектора, то в сектор, що заражається, поміщається перша частина вірусу, решта частин розміщується в інших секторах.

Щоб помістити вірус повністю в завантажувальний сектор, слід врахувати таке:

— розмір MBR вінчестера та BOOT-сектора дискети — 512 байтів (200h).

— в MBR програма-завантажувальник займає не більше 446 байтів. Розмір завантажувальника в boot-секторі дискети в різних версіях DOS різний. На завантажувальній дискеті, відформатованій під MSWIN 4.1 (Windows 98) для завантаження MS DOS 7.1 (Для завантажувальних дискет під: Windows™ 98, Windows™ 98 Second Edition, Windows™ ME і навіть Windows™ XP цей **Boot Record** містить **один і той самий код!**), власне програма-завантажувальник займає 321 байт (від адреси 003Eh до адреси 017Eh включно (виходить 383 — 62 = 321)). Потім ідуть 4 байти регістрів даних, потім — три повідомлення про помилку (“Invalid system disk”, “Disk I/O error”, “Replicate the disk, and then press any key”), які сукупно займають 84 байти (від адреси 0183h до адреси 01D6h включно). Потім ідуть назви двох системних файлів IO.SYS та MSDOS.SYS і ще три байти даних (всього 25 байтів).[1] Два останніх байта boot- та MBR-секторів містять код 55AAh. Якщо його затерти, система перестане з цього диска завантажуватися.

Таким чином, розмір коду вірусу не повинен перевищувати 434 байтів. Якщо вдасться цього досягти, звертання до диску відбуватиметься коректно, і навіть така програма, як NDD, не фіксуватиме змін в boot- чи MBR-секторах, що вельми важливо.

Існує кілька способів, якими віруси записують оригінальній MBR та своє продовження на диску: в сектори вільних кластерів логічного диска, в невикористовувані або рідко використовувані сектори диска або в сектори за межами логічного диска. В першому випадку ці сектори помічаються як збійні. В другому випадку використовуються або сектори між MBR та першим boot-сектором, або останні сектори вінчестерів.

Збереження за межами диска виконується двома способами. Перший полягає в тому, що вірус змінює записи в Disk Partition Table вінчестера або BPB (BIOS Partition Block) дискети, зменшуючи обсяг логічного диска, і записує себе у вільні, невидимі для системи сектори. Другий – запис за межами фізичного розбиття диска. На дискетах такі віруси форматують собі 40-й чи 80-й трек (зазвичай там їх 0-39 або 0-79). Вірус Hare записується за межами доступного простору вінчестера (якщо обладнання дозволяє).

Є і інші методи. Наприклад, віруси сімейства Azusa містять в своєму тілі стандартний завантажувач MBR, тому просто записуються на оригінальний без його збереження.

Завантажувальні віруси були широко розповсюджені в еру MS-DOS, однак в останні роки подібні атаки відбувалися дуже нечасто. Але в 2005 році спеціалісти з eEye Digital Security в своїй доповіді на конференції Black Hat (впливові конференції з питань комп’ютерної безпеки) показали можливість приховання rootkit (до них дійдемо) за допомогою запису до MBR.

Як стверджують спеціалісти з Verisign iDefense Intelligence Team, в грудні 2007 з’явився новий різновид шкідливих програм, які використовують давно забутий спосіб автозавантаження. Виявлений ними троянець Mebroot створено з використанням коду, продемонстрованого на Black Hat. Троянець встановлюється на перший сектор жорсткого диска, до MBR, завантажується при його зчитуванні в пам’ять, після чого вносить зміни в ядро Windows так, щоб ускладнити своє виявлення антивірусним ПЗ.

[1] http://mirror.href.com/thestarman/asm/mbr/WIN98FDB.htm

За даними iDefense, починаючи з середини грудня зловмисникам вдалося заразити за допомогою Trojan.Mebroot близько 5 тис. комп'ютерів. Основний шлях зараження – через шкідливі сайти.

Макро-віруси

Найбільшого поширення набули макро-віруси для Microsoft Office (Word, Excel і PowerPoint), що зберігають інформацію у форматі OLE2 (Object Linking and Embedding). Віруси в інших застосуваннях достатньо рідкісні. Фізичне розташування вірусу усередині файлу MS Office залежить від його формату, який у разі продуктів Microsoft надзвичайно складний, — кожен файл-документ Word, Office97 або таблиця Excel є послідовністю блоків даних (кожен з яких також має свій формат), об'єднаних між собою за допомогою великої кількості службових даних. Унаслідок такої складності форматів файлів Word, Excel і Office97 представити розташування макро-вірусу у файлі можна лише схематично:

Незаражений файл-документ або таблиця

Заголовок файлу
Службові дані (каталоги, FAT)
Текст
Шрифти
Макроси (якщо є)
Інші дані

Вірус в файлі-документі або таблиці

Заголовок файла
Службові дані (каталоги, FAT)
Текст
Шрифти
Макроси (якщо є)
Макроси віруса
Інші дані

При роботі з документами і таблицями MS Office виконує різні дії: відкриває документ, зберігає, друкує, закриває і т.д. При цьому MS Word, наприклад, шукає і виконує відповідні «вбудовані макроси» — при збереженні файлу по команді File/Save викликається макрос FileSave, при збереженні по команді File/SaveAs — FileSaveAs, при друці документів — FilePrint і т.д., якщо, звичайно, такі макроси визначені.

Існує також декілька «автомакросів», що автоматично викликаються за різних умов. Наприклад, при відкритті документа MS Word перевіряє його на наявність макросу AutoOpen. Якщо такий макрос присутній, то Word виконує його. При закритті документа Word виконує макрос AutoClose, при запуску Word викликається макрос AutoExec, при завершенні роботи — AutoExit, при створенні нового документа — AutoNew. Автоматично (тобто без участі користувача) виконуються також макроси/функції, що асоціюються з якою-небудь клавішею, моментом часу або датою.

Макро-віруси, що вражають файли MS Office, як правило, користуються одним з перерахованих вище прийомів — у вірусі або присутнє авто-макрос (авто-функція), або перевизначений один із стандартних системних макросів (асоційований з яким-небудь пунктом меню), або макрос вірусу викликається автоматично при натисненні на яку-небудь клавішу або комбінацію клавіш. Отримавши управління макро-вірус переносить свій код в інші файли, зазвичай у файли, які редагуються в даний момент. Рідше макро віруси самостійно шукають інші файли на диску.

Таким чином, віруси використовують один з трьох прийомів. 1) містять в своєму тексті авто-функцію, 2) перевизначають стандартний системний макрос, зв'язаний з певним пунктом меню, 3) викликаються при натисканні певної клавіші або їх комбінації.

При запуску віруси переносять свій код у область глобальних макросів документа, які після закриття документа автоматично записують в DOT-файл глобальних макросів, і завантажуються при кожному завантаженні Word.

Іноді віруси проникають в систему під виглядом або в складі add-in файлів і зберігаються там, не змінюючи Normal.dot.

3. Способи маскування вірусів

1. Латентний період (активація через деякий час та/або за певних умов).
2. „Стелс”
3. Поліморфізм
4. Метаморфізм
5. Маскування макровірусів

1. Латентний період — активація вірусу через певний проміжок часу або за певних умов. Прикладом є вірус CIH („Чорнобиль”), написаний Chen Ing Hau.

2. «Стелс».
На прикладі OneHalf:

За допомогою власного обробника переривання Int12h (визначення розміру використовуваної пам’яті) вірус приховує зменшення розміру вільної пам'яті на 4 кілобайти для всіх програм, окрім програми CHKDSK (утиліта перевірки диска з комплекту DOS) і програми Norton Commander. За допомогою власного обробника переривання обслуговування DOS Int21h приховує приріст довжини зараженого файлу. При спробі трасування вірусу в пам'яті за допомогою виклику переривання Int1h (покрокове переривання для відлагодження та трасування) він за допомогою власного обробника «завішує» систему організацією циклу без виходу.

3. Поліморфізм.

Поліморфізм — техніка, що дозволяє утруднити виявлення комп'ютерного вірусу за допомогою скан-рядків і, можливо, евристики. Вірус, що використовує таку техніку, називається поліморфним.

Шифрування - найчастіше використовуваний метод досягнення поліморфізму в коді. Проте, не весь код можна зашифрувати, оскільки в цьому разі вірус перестане працювати. Маленька частина залишається незашифрованою і використовується для запуску зашифрованого коду. Антивірусне програмне забезпечення націлюється саме цю невелику ділянку незашифрованого коду. Програмісти-зловмисники прагнули захистити свій поліморфний код від цієї антивірусної стратегії перезаписом незашифрованого механізму розшифровки кожного разу, коли вірус або черв'як заражає новий файл або комп’ютер.

Поліморфізм полягає у формуванні коду вірусу «на льоту» — вже під час виконання, при цьому сама процедура, що формує код також не повинна бути постійною і видозмінюється при кожному новому зараженні. Більшість антивірусних програм намагаються виявити шкідливий код, що відповідає комп'ютерному вірусу, або частину коду такого вірусу, за допомогою перевірки файлів і даних, що знаходяться на комп'ютері або пересилаються через комп'ютерну мережу або Інтернет. Якщо змінити код вірусу, що відповідає за пошук і зараження нових файлів, або яку-небудь іншу важливу його частину, то антивірус не зможе виявити такий "змінений" вірус. Часто код вірусу "міняють", додаючи оператори NOP або інші оператори, що не змінюють алгоритм. Таким чином, виявлення по-справжньому поліморфних вірусів методом скан-рядків неможливе.

Після появи поліморфізму антивірусні продукти, в свою чергу, також опанували нові методи: евристика (складний сигнатурний аналіз для виявлення основних сигнатур в різних мутаціях дешифрувального механізму задля більш надійного виявлення цих шкідливих програм) и емулятори коду (це означає, що файл запускається в захищеному ізольованому середовищі віртуального комп’ютера, вбудованого в анти-вірус. Файл аналізується на предмет поведінки, характерної для вірусів, наприклад, спроб знайти інші виконувані файли та модифікувати їх).

4. Метаморфізм

В комп'ютерній вірусології **метаморфний код** — це код, здатний до самоперепрограмування. Найчастіше він це робить шляхом переведення свого коду в яке-небудь тимчасове представлення, редагування цього представлення, а потім переведення його назад в двійковий код. Ця процедура застосовується до всього вірусу, включаючи механізм метаморфізму. Вона запускається тоді, коли вірус збирається заразити нові файлі, тому вірус-нащадок ніколи не виглядатиме так, як його батько.

Метаморфний код більш ефективний та стійкий за поліморфний, оскільки більшість сучасних антивірусних програм аналізує код на шкідливість навіть під час виконання цього коду.

Інше значення терміну „метаморфізм" — здатність вірусу заражати виконавчі файли двох або більше ОС або навіть існувати на комп'ютерах з різною архітектурою. Такий вірус часто являє собою сукупність різних окремих вірусів. Початковий код вірусу пишеться так, щоб він коректно транслювався в машинний код всіх платформ, на яких має працювати. Теоретично, метаморфний вірус здатен перетворити своє тимчасове представлення в набір інструкцій для абсолютно іншої комп'ютерної архітектури.

Метаморфізм означає, що мутує весь код, а не лише шифратор/дешифратор, як у випадку поліморфізму. Втім, такі віруси не дуже розповсюджені, оскільки їх дійсно дуже складно написати.

≈90% коду вірусу займає механізм метаморфізму, а його „бойова" частина — що залишилося. Структура виглядає так:

Метаморфний механізм	Власне вірус

Типова структура метаморфного механізму:

Дизасемблер	Стискач	Мутатор	Розширювач	Асемблер

— *Дизасемблер*. Перша частина механізму. Декодує кожну інструкцію для визначення її довжини, використовуваних регістрів та іншої інформації. Повинен вміти декодувати такі інструкції, як JMP та CALL, що змінюють регістр IP.

— *Стискач*. Стискає дизасембльований код, згенерований в попередньому поколінні. Слугує для запобігання експоненційного зростання довжини вірусу з кожним наступним поколінням. Найскладніша частина вірусу, хоча складність безпосередньо залежить від того, які механізми маскування застосовуються автором.

— *Мутатор.* Основна частина механізму. На цьому етапі відбувається зміна коду.

— *Розширювач*. Стискач навпаки — кодує одну інструкцію кількома, результат роботи яких той самий, або додає „порожній" код, який ніяк не впливає на роботу алгоритму, але створює додаткові перешкоди антивірусу.

— *Асемблер*. Перетворює те, що побудував розширювач, в двійковий код, одночасно відновлюючи оператори JMP, CALL та інші.[2]

5. Маскування макровірусів.

Три підходи до маскування макро-вірусів, що базуються на здатності макросів створювати, редагувати та виконувати код інших макросів. Такі віруси мають поліморфний макрос-завантажувач, який викликає редактор макросів, створює новий макрос, заповнює кодом базового макросу, запускає на виконання і потім, як правило, знищує. Основний код міститься або в коді самого макросу як простий текст, або в області змінних документа, або в області auto-text.

Лекція 3
1. Класифікація антивірусних програм

1. Класифікація антивірусних пакетів

Для виявлення, видалення і захисту від комп'ютерних вірусів розроблено декілька видів спеціальних програм, які дозволяють виявляти і знищувати віруси. Такі програми називаються антивірусними. Розрізняють наступні види антивірусних програм:

— програми-детектори
— програми-доктори або фаги
— програми-ревізори
— програми-фільтри
— програми-вакцини або імунізатори

Програми-детектори

Детектори здійснюють пошук характерної для конкретного вірусу сигнатури в оперативній пам'яті і у файлах і при виявленні видають відповідне повідомлення. Недоліком таких антивірусних програм є те, що вони можуть знаходити тільки ті віруси, які відомі розробникам таких програм.

Програми-доктори або фаги

Фаги не тільки знаходять заражені вірусами файли, але і «лікують» їх, тобто видаляють з файлу тіло програми-вірусу, повертаючи файли в початковий стан. На початку своєї роботи фаги шукають віруси в оперативній пам'яті, знищуючи їх, і тільки тоді переходять до «лікування» файлів. Серед фагів виділяють поліфаги, тобто програми-доктори, призначені для пошуку і знищення великої кількості вірусів. Найбільш відомі з них: Norton AntiVirus, Doctor Web, NOD32, AVK. Враховуючи, що постійно з'являються нові віруси, програми-детектори і програми-доктори швидко застарівають, і потрібне регулярне оновлення версій.

Антівіруси-поліфаги – найбільш поширені засоби по боротьбі з шкідливими програмами. Історично вони з'явилися першими і до цих пір утримують безперечне лідерство в цій області.

В основі роботи поліфагів лежить простій принцип – пошук в програмах і документах знайомих ділянок вірусного коду (так званих сигнатур вірусів). Під сигнатурою можуть розумітися різні речі. У загальному випадку сигнатура – це такий запис про вірус, який дозволяє однозначно ідентифікувати присутність вірусного коду в програмі або документі. Найчастіше сигнатура – це безпосередньо ділянка вірусного коду або його контрольна сума (дайджест).

Спочатку антівіруси-поліфаги працювали за дуже простим принципом – здійснювали послідовний перегляд файлів на предмет знаходження в них вірусних програм. Якщо сигнатура вірусу була виявлена, то проводилася процедура видалення вірусного коду з тіла програми або документа. Перш ніж почати перевірку файлів, програма-фаг завжди перевіряє оперативну пам'ять. Якщо в оперативній пам'яті виявляється вірус, то відбувається його деактивація. Це викликано тим, що часто вірусні програми проводять зараження тих програм, які запускаються або відкриваються в той момент, коли вірус знаходиться в активній стадії (це пов'язано з прагненням економити на зусиллях по пошуку об'єктів зараження). Таким чином, якщо вірус залишиться активним в пам'яті, то тотальна перевірка всіх виконуваних файлів приведе до тотального зараження системи.

Наразі вірусні програми значно ускладнилися. Наприклад, з'явилися так звані “stealth-віруси”. В основі їх роботи лежить той факт, що операційна система при зверненні до периферійних пристроїв (у тому числі і до жорстких дисків) використовує механізм переривань. Механізм переривань працює так. При виникненні переривання управління передається спеціальній програмі – ”обробникові переривання”. Ця програма відповідає за введення і виведення інформації у чи з периферійного пристрою. Крім того, переривання діляться на рівні взаємодії з периферією (у нашому випадку – з жорсткими і гнучкими дисками). Є рівень операційної системи (у середовищі MS DOS – переривання 25h), є рівень базової системи введення/виводу (рівень BIOS – переривання 13h). Досвідчені системні програмісти можуть працювати і безпосередньо, звертаючись до портів введення/виводу пристроїв. Але це же досить серйозне і важке завдання. Така багаторівнева система зроблена, перш за все, з метою збереження переносимості застосувань. Саме завдяки такій системі, скажімо, виявилося можливим здійснювати запуск DOS-застосувань в багатозадачних середовищах типу MS Windows або IBM OS/2.

Але в такій системі спочатку прихована і уразливість: управляючи обробником переривань, можна управляти потоком інформації від периферійного пристрою до користувача. Stealth-віруси, зокрема, використовують механізм перехоплення управління при виникненні переривання. Замінюючи оригінальний обробник переривання своїм кодом, stealth-віруси контролюють читання даних з диска. У випадку, якщо з диска читається заражена програма, вірус “викушує” власний код (зазвичай код не буквально ”викушується”, а відбувається підміна номера читаного сектора диска). У результаті користувач отримує для читання “чистий” код. Таким чином, до тих пір, поки вектор обробника

переривань змінений вірусним кодом, сам вірус активний в пам'яті комп'ютера, виявити його простим читанням диска засобами операційної системи неможливо. Схожий механізм маскування використовується і завантажувальними вірусами.

В цілях боротьби зі stealth-вірусами рекомендується здійснювати альтернативне завантаження системи з зовнішнього носія і лише після цього проводити пошук і видалення вірусних програм.

Зважаючи на все вищесказане, антівіруси-поліфаги виявляються максимально ефективними тільки при боротьбі з вже відомими вірусами, тобто з такими, чиї сигнатури і методи поведінки знайомі розробникам. Тільки в цьому випадку вірус зі 100-процентною точністю буде виявлений і видалений з пам'яті комп'ютера, а потім – і з усіх файлів, що перевіряються. Якщо ж вірус невідомий, то він може достатньо успішно протистояти спробам його виявлення і лікування. Тому головне при користуванні будь-яким поліфагом – якомога частіше оновлювати версії програми і вірусні бази. Для зручності користувачів бази винесені в окремий модуль, і користувачі можуть оновлювати ці бази щодня за допомогою Інтернету.

Осібно тут стоять так звані евристичні аналізатори. Річ у тому, що існує велика кількість вірусів, алгоритм яких практично скопійований з алгоритму інших вірусів. Як правило, такі варіації створюють непрофесійні програмісти, які з якихось причин вирішили написати вірус. Для боротьби з такими “копіями” і були придумані евристичні аналізатори. З їх допомогою антивірус здатний знаходити подібні аналоги відомих вірусів, повідомляючи користувача, що у нього, схоже, завівся вірус. Природно, надійність евристичного аналізатора не 100%, але все таки його коефіцієнт корисної дії більше 0,5. Віруси, які не розпізнаються антивірусними детекторами, здатні написати тільки найбільш досвідчені і кваліфіковані програмісти.

Евристичним аналізатором коду називається набір підпрограм, що аналізують код виконуваних файлів, пам'яті або завантажувальних секторів для виявлення в ньому різних типів комп'ютерних вірусів. Основною частиною евристичного аналізатора є емулятор коду. Емулятор коду працює в режимі перегляду, тобто його основне завдання – не просто емулювати роботу програми, а виявляти всілякі події, тобто ділянку коду чи виклик певної функції операційної системи, направлені на перетворення системних даних, роботу з файлами або часто використовувані вірусні конструкції. Грубо кажучи, емулятор проглядає код програми і виявляє ті дії, які ця програма здійснює. Якщо дії цієї програми укладаються в якусь певну схему, то робиться висновок про наявність в програмі вірусного коду.

Звичайно, вірогідність як пропуску, так і помилкового спрацьовування вельми висока. Проте правильно використовуючи механізм евристики, користувач може самостійно прийти до вірних виводів. Наприклад, якщо антивірус видає повідомлення про підозру на вірус для одиничного файлу, то вірогідність помилкового спрацьовування вельми висока. Якщо ж таке повторюється на багатьох файлах (а до цього евристик нічого підозрілого в цих файлах не виявляв), то можна говорити про зараження вашої системи вірусом з вірогідністю, близькою до 100%.

Використання евристичного аналізатора, крім всього вищепереліченого, дозволяє також боротися з вірус-генераторами і поліморфними вірусами. Класичний метод з виявленням вірусів за сигнатурою в цьому випадку взагалі виявляється неефективним. Вірус-генератори – це спеціалізований набір бібліотек, який дозволяє легко сконструювати свій власний вірус, навіть маючи слабкі пізнання в програмуванні. Написавши нескладну програму, ви далі підключаєте до цієї програми бібліотеки генератора, вставляєте в потрібних місцях виклики зовнішніх процедур – і ось ваш елементарний вірус перетворився на достатньо складний продукт. Найсумніше, що в цьому випадку сигнатура вірусу буде кожного разу інша, тому відстежити вірус виявляється можливим тільки по характерних викликах зовнішніх процедур – а це вже робота евристичного аналізатора. Поліморфний вірус має ще складнішу структуру. Само тіло вірусу видозмінюється від зараження до зараження, при цьому зберігаючи своє функціональне наповнення.

В простому випадку – якщо розкидати в тілі вірусу випадковим чином „порожні” оператори (типу “ mov ax, ax” або “nop”), які нічого, по суті, не роблять, то тіло вірусного коду зазнає значних змін, а алгоритм залишиться тим самим. В цьому випадку на допомогу також приходить евристичний аналізатор.

Іншим засобом боротьби з вірусами є аналіз поведінки, покликаний ліквідувати вроджену слабкість антивірусних пакетів, а саме залежність ефективності їх роботи від обсягу бази відомих сигнатур вірусів. Компанії виробники не здатні швидко реагувати на появу нових вірусів, бо нова шкідлива програма з’являється в середньому кожні 30 секунд, а навіть найоперативніші антивіруси оновлюються від 60 до 600 разів на місяць (безумовний лідер – Norton IS 2009 з 6202 оновленнями). Цей механізм вбудовано у всі найвідоміші і найпопулярніші антивіруси.

Програми-ревізори

Відносяться до найнадійніших засобів захисту від вірусів. Ревізори запам'ятовують початковий стан програм, каталогів і системних областей диска тоді, коли комп'ютер не заражений вірусом, а потім періодично або за бажанням користувача порівнюють поточний стан з результатним. Виявлені зміни виводяться на екран монітора. Як правило, порівняння станів проводять відразу після завантаження операційної системи. При порівнянні перевіряються довжина файлу, код циклічного контролю (контрольна сума файлу), дата і час модифікації, інші параметри. Програми-ревізори мають достатньо розвинені алгоритми, виявляють стелс-віруси і можуть навіть очистити зміни версії програми, що перевіряється, від змін, внесених вірусом. До програм-ревізорів належить широко поширена в Росії програма Adinf від «Діалогнаука». Недолік — сильно гальмують роботу комп'ютера.

Антивірусні програми-ревізори дозволяють виявити вірус. Найчастіше виявленням вірусу справа і закінчується. Існує блок лікування для популярного антивіруса-ревізора Adinf, так званий Cure Module, але такий блок дозволяє лікувати лише ті файли, які були не заражені на момент створення бази даних програми. Проте виявити вірус на комп'ютері (або принаймні запідозрити його наявність) антивіруси-ревізори можуть з великим ступенем надійності. Зазвичай найоптимальнішою є зв'язка поліфаг і ревізор. Ревізор служить для виявлення факту зараження системи. Якщо система заражена, то в справу пускається поліфаг. Якщо ж йому не вдалося знищити вірус, то можна звернутися до розробника антивірусних засобів – швидше за все на ваш комп'ютер потрапив новий, невідомий розробникам вірус.

Основу роботи ревізорів складає контроль за змінами, характерними для роботи вірусних програм. Далі ми розглянемо, як цей контроль здійснюється. При установці програми створюються спеціальні таблиці. У них міститься інформація: про контрольні суми незмінних файлів, вміст системних областей, адреси обробників переривань, розмір доступної оперативної пам'яті і т.п. Далі робота ревізора полягає в порівнянні поточного стану диска з раніше збереженими даними, тому украй важливо, щоб всі контрольні таблиці створювалися не на зараженій машині. Тільки в цьому випадку робота ревізора буде достатньо ефективною. Отже, перейдемо до стадій роботи програми-ревізора.

Контроль оперативної пам'яті. Ця стадія перевірки включає процедури виявлення слідів активних завантажувальних і stealth-вірусів в пам'яті комп'ютера. Якщо такі алгоритми будуть знайдені, ви отримаєте відповідне попередження. Спочатку програма шукає вже знайомі віруси. Далі програма перевіряє, чи змінився обробник Int13h. Якщо він змінився, то з вірогідністю 90% можна сказати, що комп'ютер інфікований завантажувальним вірусом (завантажувальні віруси вимушені перехоплювати це переривання з тим, щоб після своєї активізації передати управління "нормальному" завантажувальному сектору і система завантажилася без збоїв). Ревізор видасть вам попередження про це і повідомить адресу в пам'яті, по якій знаходиться новий обробник Int13h. В принципі інформація про місцезнаходження обробника необхідна програмістам і системним адміністраторам, а рядовому користувачеві слід звернути увагу на попередження. Навіть якщо ревізор встановлюється вже на заражений вірусом комп'ютер і реальна адреса обробника Int13h маскується вірусом, в 85% випадків ревізорові вдається виявити дійсну адресу обробника Int13h в BIOS і працювати, використовуючи його. Якщо з яких-небудь причин ревізорові не вдалося отримати реальну адресу обробника, то видається попередження. Дійсна адреса обробника переривання досягається шляхом покрокового перегляду тіла вірусу (по алгоритму своєї роботи завантажувальний вірус вимушений врешті-решт передавати управління оригінальному обробникові). Деякі віруси блокують трасування переривань: при спробі трасувати їх коди вони "завішують" систему, перезавантажують комп'ютер і т.д. Тому, якщо при трасуванні переривань комп'ютер починає поводитися "дивно", то слід бути дуже обережним – не виключено, що оперативна пам'ять уражена вірусом.

Важливим параметром є і розмір вільної оперативної пам'яті. Зазвичай ревізор запускається найпершим, до завантаження яких-небудь ще програм. Якщо ж розмір оперативної пам'яті зменшився – це вірна ознака присутності в ОЗУ ще якоїсь програми. Швидше за все програма ця – вірус.

Контроль системних областей. Контроль системних областей призначений для виявлення вірусів, які використовують для своєї активації механізм завантаження. Як відомо, першою з диска завантажується завантажувальний запис (boot record), який містить в собі міні-програму, що управляє подальшим завантаженням. Для жорсткого диска першої проводиться завантаження головного завантажувального запису (MasterBootRecord або MBR).

Програми-ревізори не можуть судити про початкову "чистоту" оперативної пам'яті, тому читання Master Boot Record відбувається трьома різними способами:

- (bios) – прямим зверненням в BIOS;
- (i13h) – читанням через BIOS-переривання Int13h;
- (i25h) – читанням засобами операційної системи (переривання Int25h).

Якщо зчитана інформація не співпадає, в наявності дія stealth-алгоритмів. Для більшої надійності читання MBR проводиться через порти жорсткого диска. На сьогодні в "дикій природі" не зустрічалися віруси, які можуть маскуватися від ревізора, що володіє такою функцією. Аналогічним чином проводиться перевірка і простого (не головного) завантажувального сектора. Зазвичай за

рахунок того, що ревізор зберігає резервну копію системних областей, відновлення пошкоджень від завантажувального вірусу відбувається досить прозаїчно: якщо користувач дає на те свою згоду, ревізор просто записує системні області наново, використовуючи збережені раніше дані.

Контроль незмінних файлів. Остання стадія перевірки, направлена на виявлення діяльності файлових вірусів – контроль зміни файлів. Для всіх файлів, які активно використовуються і в той же час не повинні змінюватися (звичайно це програми типу win.com і т. п.) створюються контрольні таблиці. У них містяться значення контрольних сум і розмірів файлів. Потім, в ході подальшого використання ревізора, інформація з дисків порівнюється з еталонною, такою, що зберігається в таблицях. Якщо інформація не співпадає, то мабуть знаходження в системі файлового вірусу. Найвиразніша ознака – зміна розміру або вмісту файлу без зміни дати створення файлу.

В принципі, рекомендується внести до розряду “незмінних” ті виконувані файли, шлях до яких вказаний в змінній PATH. Вони найчастіше стають жертвою файлових вірусів.

Щоб не дати stealth-вірусам “обдурити” систему, читання даних також відбувається як засобами операційної системи, так і засобами BIOS. Якщо ці дані не співпали, то можна говорити про те, що в системі активно діє вирус-“невидимка”.

Після того, як всі файли перевірені, ревізори часто зберігають додаткові області пам'яті, які можуть бути зіпсовані вірусами. Це FLASH- і CMOS-пам'ять. Ці області пам'яті також змінюються достатньо рідко і тому їх зміни можуть бути підозрілі.

Програми-фільтри або «сторожі»

Це невеликі резидентні програми, призначені для виявлення підозрілих дій при роботі комп'ютера, характерними для вірусів. Такими діями можуть бути: спроби корекції файлів з розширеннями COM, EXE зміна атрибутів файлу прямий запис на диск за абсолютною адресою запис в завантажувальні сектори диска завантаження резидентної програми При спробі якої-небудь програми провести вказані дії «сторож» посилає користувачеві повідомлення і пропонує заборонити або вирішити відповідну дію. Програми-фільтри вельми корисні, оскільки здатні виявити вірус на найранішій стадії його існування до розмноження. Проте, вони не «лікують» файли і диски. Для знищення вірусів потрібно застосувати інші програми, наприклад фаги. До недоліків програм-сторожів можна віднести їх «настирливість»(наприклад, вони постійно видають попередження про будь-яку спробу копіювання виконуваного файлу), а також можливі конфлікти з іншим програмним забезпеченням. Прикладом програми-фільтру є програма Vsafe, що входить до складу пакету утиліт MS DOS, та ZoneAlarm.

Вакцини або імунізатори.

Це резидентні програми, що запобігають зараженню файлів. Вакцини застосовують, якщо відсутні програми-доктори, що «лікують» цей вірус. Вакцинація можлива тільки від відомих вірусів. Вакцина модифікує програму або диск так, щоб це не відбивалося на їх роботі, а вірус сприйматиме їх зараженими і тому не упровадиться. В даний час програми-вакцини мають обмежене застосування. Своєчасне виявлення заражених вірусами файлів і дисків, повне знищення виявлених вірусів на кожному комп'ютері дозволяють уникнути розповсюдження вірусної епідемії на інші комп'ютери.

Деякі вакцини здатні здійснювати активніші дії. Наприклад, вакцина VS_ONE_H, написана спеціально для боротьби з вірусом OneHalf ялтинським програмістом А.Крижанівським, також перехоплює int13h, і використовує алгоритм антитрасування, що називається в программістському світі "поплавець". Як тільки хто-небудь в системі здійснював спробу відтрасувати переривання 13h, вакцина видає попередження і зупиняє систему.

Метод «поплавця» заснований на тому, що трасування робиться покроково – процесор перемикається в режим відладки, і після виконання кожної команди викликає debug-переривання, використовуване, зазвичай відладчиками для покрокової відладки програми, щоб проаналізувати результати виконання команди. Виклик налагоджувального переривання має на увазі, що використовується стек процесора, в якому зберігається адреса повернення. Це і використовується для того, щоб визначити трасування.

Ми забороняємо переривання, потім поміщаємо в перший вільний осередок стека код 0000 або FFFF (обидві адреси не можуть бути адресою повернення), потім прочитуємо з першого вільного осередку стека значення, здійснюємо переривання і порівнюємо лічене значення з тим, що ми туди поміщали.

Сенс методу в тому, що коли програма виконується нормально, після заборони переривань, ніхто не зможе перервати виконання наший програми, і тому, помістивши в перший вільний осередок стека яке-небудь значення, ми потім зможемо його ж звідти і прочитати, тому що стек використовувати нікому. Якщо ж програма виконується покроково, то, не дивлячись на заборону переривань, процесор все одно викличе debug-переривання після команди, якою ми поміщали в стек число. Це debug-переривання помістить у

використаний нами осередок стека адресу повернення, після чого ця адреса ми і прочитаємо наступною командою читання із стека. Отримавши неспівпадання записаного і зчитаного чисел ми зможемо переконатися в тому, що включений режим покрокового виконання команд процесора і ми можемо стверджувати, що піддалися зараженню вірусом.

Суміжні області антивірусного захисту та забезпечення безпеки комп'ютерів та мереж

DoS-атака (від англ. Denial of Service — «відмова в обслуговуванні») і DDoS-атака (Distributed Denial of Service — «розподілена відмова обслуговування») — це різновиди атак зловмисників на комп'ютерні системи. Метою цих атак є створення таких умов, при яких легітимні (правомочні) користувачі системи не можуть одержати доступ до ресурсів, що надаються системою, або цей доступ утруднений.

Види DoS-атак

Існують різні причини, з яких може виникнути DoS-умова:

- **Помилка в програмному коді**, що приводить до звернення до невживаного фрагмента адресного простору, виконання неприпустимої інструкції або іншої необроблюваної виняткової ситуації, коли відбувається аварійне завершення серверного застосування. Класичним прикладом є звернення за нульовим (англ. null) покажчиком.
- **Недостатня перевірка даних користувача**, що приводить до нескінченного або тривалого циклу або підвищеного тривалого споживання процесорних ресурсів (вичерпанню процесорних ресурсів) або виділення великого об'єму оперативній пам'яті (вичерпанню пам'яті).
- **Флуд** (англ. flood) — атака, пов'язана з великою кількістю зазвичай безглуздих або сформованих в неправильному форматі запитів до комп'ютерної системи або мережевого устаткування, що має за мету або призвела до відмови в роботі системи через вичерпання ресурсів системи — процесора, пам'яті або каналів зв'язку.
- **Атака другого роду** — атака, яка прагне викликати помилкове спрацьовування системи захисту і таким чином привести до недоступності ресурсу.

Якщо атака (зазвичай флуд) проводиться одночасно з великої кількості IP-адрес, то в цьому випадку вона називається розподіленою атакою на відмову в обслуговуванні (DDoS).

Виявлення DoS-атак

Існує думка, що спеціальні засоби для виявлення DoS-атак не потрібні, оскільки факт DoS-атаки неможливо не помітити. У багатьох випадках це дійсно так. Проте достатньо часто відбувалися успішні атаки, які були помічені жертвами лише через 2-3 доби. Бувало, що негативні наслідки атаки (типу флуд) полягали в зайвих витратах на оплату трафіку, що з'ясовувалося лише при отриманні рахунку. Крім того, багато методів виявлення атак неефективні поблизу об'єкта атаки, але ефективні на магістральній мережі. У такому разі доцільно ставити системи виявлення саме там, а не чекати, поки користувач, що піддався атаці, сам її зафіксує та звернеться по допомогу. До того ж, для ефективної протидії необхідно знати тип, характер і інші показники DoS-атаки, а оперативно отримати ці відомості якраз і дозволяють системи виявлення.

Методи виявлення можна розділити на декілька великих груп:

- сигнатурні — засновані на якісному аналізі трафіку;
- статистичні — засновані на кількісному аналізі трафіку;
- гібридні — поєднуючі в собі достоїнства двох попередніх методів.

Захист від DoS-атак

Заходи протидії DoS-атакам можна розділити на пасивні й активні, а також на превентивні та реакційні.

Нижче наведено короткий перелік основних методів.

- **Запобігання**. Профілактика причин, що спонукають тих або інших осіб організовувати DoS-атаки. Дуже часто атаки є наслідком особистої образи, політичних, релігійних розбіжностей і т.п.
- **Фільтрація (пропускання «своїх» та блокування «чужих» пакетів) і блекхолінг (відстежування пакетів, що надходять з адреси, що належить простору ще не присвоєних адрес)**. Ефективність цих методів знижується

мірою наближення до об’єкта атаки і підвищується мірою наближення до її джерела.

- **Усунення вразливостей**. Не працює проти атак типу флуд, для яких «вразливістю» є скінченність тих або інших ресурсів.
- **Нарощування ресурсів (щоб на довше вистачило).**
- **Роззосередження.** Побудова розподілених і продубльованих систем, які не припинять обслуговувати користувачів, навіть якщо деякі їх елементи стануть недоступні через атаку.
- **Ухилення.** Відведення безпосередньої мети атаки (доменного імені або IP-адреси) подалі від інших ресурсів, які часто також піддаються нападу разом з безпосередньою метою.
- **Активні дії у відповідь**. Дії на джерела, організатора або центр управління атакою. Заходи можуть бути як технічного характеру (не рекомендується), так і організаційно-правового характеру.

Троянські програми

Троянські програми розрізняються між собою за тими діями, які вони проводять на зараженому комп'ютері.

Backdoor — троянскі утиліти віддаленого адміністрування

Троянські програми цього класу є утилітами віддаленого адміністрування комп'ютерів в мережі. За своєю функціональністю вони багато в чому нагадують різні системи адміністрування, що розробляються і поширюються фірмами-виробниками програмних продуктів.

Єдина особливість цих програм примушує класифікувати їх як шкідливі троянські програми: відсутність попередження про інсталяцію і запуск. При запуску «троян» встановлює себе в системі і потім стежить за нею, при цьому користувачеві не видаються ніяких повідомлень про дії трояну в системі. Більш того, посилання на «трояни» може бути відсутнім в списку активних застосувань. В результаті «користувач» цієї троянської програми може і не знати про її присутність в системі, тоді як його комп'ютер відкритий для віддаленого управління.

Утиліти прихованого управління дозволяють робити з комп'ютером все, що в них заклав автор: приймати або посилати файли, запускати і знищувати їх, виводити повідомлення, стирати інформацію, перезавантажувати комп'ютер і т.д. В результаті цей троян може бути використаний для виявлення і передачі конфіденційної інформації, для запуску вірусів, знищення даних і т.п. — уражені комп'ютери виявляються відкритими для зловмисних дій хакерів.

Таким чином, троянські програми даного типу є одним з найнебезпечніших видів шкідливого програмного забезпечення, оскільки в них закладена можливість найрізноманітніших зловмисних дій, властивих іншим видам троянських програм.

Окремо слід зазначити групу бекдорів, здатних розповсюджуватися по мережі і упроваджуватися в інші комп'ютери, як це роблять комп'ютерні черв'яки. Відрізняє таких «троянів» від черв'яків той факт, що вони розповсюджуються по мережі не мимоволі (як черві), а тільки по спеціальній команді «господаря», керівника даної копієї троянської програми.

Trojan-PSW — крадіжка паролів

Дане сімейство об'єднує троянські програми, що «крадуть» різну інформацію із зараженого комп'ютера, зазвичай — системні паролі (PSW — Password-Stealing-Ware). При запуску PSW-трояни шукають системні файли, що зберігають різну конфіденційну інформацію (зазвичай номери телефонів і паролі доступу до інтернету) і посилають її по вказаному в коді «трояну» електронній адресі або адресам.

Існує PSW-троян, який повідомляє і іншу інформацію про заражений комп'ютер, наприклад, інформацію про систему (розмір пам'яті і дискового простору, версія операційної системи), тип використовуваного поштового клієнта, IP-адресу і т.п. Деякі трояни даного типу «крадуть» реєстраційну інформацію до різного програмного забезпечення, коди доступу до мережевих ігор і інше.

TROJAN-AOL — сімейство троянських програм, що «крадуть» коди доступу до мережі AOL (America Online). Виділені в особливу групу через свою численність.

Trojan-Clicker — інтернет-клікери

Сімейство троянських програм, основна функція яких — організація несанкціонованих звернень до інтернет-ресурсів (зазвичай до веб-сторінок). Досягається це або посилкою відповідних

команд браузеру, або заміною системних файлів, в яких вказані «стандартні» адреси інтернет-ресурсів (наприклад, файл hosts в MS Windows).

У зловмисника можуть бути наступні цілі для подібних дій:

збільшення відвідуваності яких-небудь сайтів з метою збільшення показів реклами;

організація DoS-атаки (Denial of Service) на який-небудь сервер;

залучення потенційних жертв для зараження вірусами або троянськими програмами.

Trojan-Downloader — доставка інших шкідливих програм

Троянські програми цього класу призначені для завантаження і установки на комп'ютер-жертву нових версій шкідливих програм, установки «троянів» або рекламних систем. Завантажені з інтернету програми потім або запускаються на виконання, або реєструються «Троєю» на автозавантаження відповідно до можливостей операційної системи. Дані дії при цьому відбуваються без відома користувача.

Інформація про імена і розташування завантажуваних програм міститься в коді і даних Трої або викачується Троєю з інтернет-ресурсу, що «управляє» (зазвичай з веб-сторінки).

Trojan-Dropper — інсталятори інших шкідливих програм

Троянські програми цього класу написані з метою прихованої інсталяції інших програм і практично завжди використовуються для «підсовування» на комп'ютер-жертву вірусів або інших троянських програм.

Дані трояни зазвичай без яких-небудь повідомлень (або з помилковими повідомленнями про помилку в архіві або невірній версії операційної системи) скидають на диск в який-небудь каталог (у корінь диска C:, у тимчасовий каталог, в каталоги Windows) інші файли і запускають їх на виконання. Зазвичай структура таких програм наступна:

Основний код
Файл 1
Файл 2
...

«Основний код» виділяє з свого файлу решту компонентів (файл 1, файл 2 ...), записує їх на диск і відкриває їх (запускає на виконання). Зазвичай один (або більш) компонентів є троянськими програмами, і як мінімум один компонент є «обманкою»: програмою-жартом, грою, картинкою або чимось подібним. «Обманка» повинна відвернути увагу користувача і/або продемонструвати те, що файл, що запускається, дійсно робить щось «корисне», тоді як троянська компонента інсталюється в систему.

В результаті використання програм даного класу хакери досягають двох цілей:

скритна інсталяція троянських програм і/або вірусів;

захист від антивірусних програм, оскільки не все з них в змозі перевірити всі компоненти усередині файлів цього типу.

Trojan-Proxy — троянські проксі-сервери

Сімейство троянських програм, що скритно здійснюють анонімний доступ до різних інтернет-ресурсів. Зазвичай використовуються для розсилки спаму. Перетворюють заражений комп’ютер на проксі-сервер, від імені якого працює зловмисник.

Trojan-Spy — шпигунські програми

Дані Трояни здійснюють електронне шпигунство за користувачем зараженого комп'ютера: інформація, що вводиться з клавіатури, знімки екрану, список активних застосувань і дії користувача з ними зберігаються в який-небудь файл на диску і періодично відправляються зловмисникові. Троянські програми цього типу часто використовуються для крадіжки інформації користувачів різних систем онлайнових платежів і банківських систем.

Trojan — інші троянські програми

До даних Троянів відносяться ті з них, які здійснюють інші дії, що потрапляють під визначення троянських програм, тобто руйнування або зловмисна модифікація даних, порушення працездатності комп'ютера і інше. У даній категорії також присутні «багатоцільові» троянські програми, наприклад, ті з них, які одночасно шпигують за користувачем і надають proxy-сервіс віддаленому зловмисникові.

ArcBomb — «бомби» в архівах

Є архівами, спеціально оформленими так, щоб викликати нештатну поведінку архіваторів при спробі розархівувати дані — зависання або істотне уповільнення роботи комп'ютера або заповнення диска великою кількістю «порожніх» даних. Особливо небезпечні «архівні бомби» для файлових і поштових серверів, якщо на сервері використовується яка-небудь система автоматичної обробки вхідної інформації — «архівна бомба» може просто зупинити роботу сервера.

Зустрічаються три типи подібних «бомб»: некоректний заголовок архіву, дані, що повторюються, і однакові файли в архіві.

Некоректний заголовок архіву або зіпсовані дані в архіві можуть привести до збою в роботі конкретного архіватора або алгоритму розархивовування при розборі вмісту архіву.

Значних розмірів файл, що містить дані, що повторюються, дозволяє заархівувати такий файл в архів невеликого розміру (наприклад, 5ГБ даних упаковуються в 200КБ RAR або в 480КБ ZIP-архів).

Величезна кількість однакових файлів в архіві також практично не позначається на розмірі архіву при використанні спеціальних методів (наприклад, існують прийоми упаковки 10100 однакових файлів в 30КБ RAR або 230КБ ZIP-архіві).

Trojan-Notifier — інформування про успешну атаку

Трояни даного типу призначені для повідомлення свого «господаря» про зараження комп'ютера. При цьому на адресу «господаря» відправляється інформація про комп'ютер, наприклад, IP-адреса комп'ютера, номер відкритого порту, адреса електронної пошти і т.п. Відсилання здійснюється різними способами: електронним листом, спеціально оформленим зверненням до веб-сторінки «господаря», ICQ-повідомленням. Дані троянські програми використовуються в багатокомпонентних троянських наборах для сповіщення свого «господаря» про успішну інсталяцію троянських компонент в систему, що атакується.

Поштові віруси. Зараження відбувається після одержання від когось невідомого листа, що містить програму-«зародок», часто хитро замасковану. Файл має вигляд <ім’я>.jpg .exe, де перед «.exe» записано багато пробілів. Windows показує для таких довгих імен лише початок, тому користувач вважає надісланий файд JPEG-малюнком і запускає його, вмикаючи програму. Вірус «прописується» в системі і починає розсилати абонентам з вашої адресної книги свої копії-зародки.
Не кажучи вже про те, що поштові клієнти дозволяють приймати листи в форматі HTML, в тому числі такі, які містять скрипти і звернення до серверних програмних компонентів, причому з можливістю автозапуску скрипта в момент перегляду одержаного листа.

Віруси в соціальних мережах Вконтакте і Одноклассники.

Експлойти

Цей вид комп'ютерної гидоти виник через те, що в програмному забезпеченні завжди є які-небудь помилки. Хай не дуже великі, хай непомітні - але вони є, і в умілих руках стають страшною зброєю. Результат дії експлойта, який проникає на комп'ютер (а іноді може навіть і не проникати) із-за помилки в мережевих застосуваннях, буває різний. Наприклад, в результаті зламувана система може одержати троян або вірус, або скинути настройки, або перезавантажитися. Отже експлойти - одні з найнебезпечніших шкідливих програм, що існують на сьогоднішній день. Велика частина користувачів стикається з ними, коли зловмисники знаходять чергову уразливість в браузере (особливо цим славиться Internet Explorer, але і Mozilla Firefox з Opera теж зрідка "радують" користувачів). Посилюється справа тим, що велика частина користувачів рідко оновлює свої браузери, хоча це, мабуть, найбільш простій і дієвий спосіб боротьби з експлойтами.

Rootkit — приховування присутності в операційній системі

Поняття rootkit прийшло до нас з UNIX. Спершу це поняття використовувалося для позначення набору інструментів, вживаних для отримання прав користувача root.

Оскільки інструменти типу rootkit на сьогоднішній день «прижилися» і на інших ОС (зокрема, на Windows), то слід визнати подібне визначення rootkit морально застарілим і таким, що не відповідає реальному положенню справ.

Таким чином, rootkit — програмний код або техніка, направлена на захована присутності в системі заданих об'єктів (процесів, файлів, ключів реєстру і т.д.).

Для поведінки Rootkit в класифікації «Лабораторії Касперського» діють правила поглинання: Rootkit — сама молодша поведінка серед шкідливих програм. Тобто, якщо Rootkit-програма має троянську складову, то вона детектується як Trojan.

Rootkit — це загальна назва набору програм, задачею яких є перебирання на себе контролю за ОС від її легітимних користувачів. Зазвичай rootkit приховуватиме своє встановлення та

перешкоджатиме видаленню через підпорядкування собі стандартних засобів безпеки системи. Для цього використовуються техніки приховування запущених процесів, файлів чи даних системи від ОС. Від початку Rootkit'и використовувалися в корисних програмах, але в останні роки все частіше використовуються в malware для полегшення утримання доступу зловмисників до системи шляхом запобігання їх виявленню. Rootkit'и існують для багатьох ОС, таких як Microsoft Windows, Mac OS X, Linux та Solaris. Rootkit'и часто модифікують складові ОС чи інсталюють себе як драйвери чи модулі ядра.

Терміном *rootkit* спочатку позначали набір прекомпільованих Unix-інструментів, таких як ps, netstat, w та passwd, які ретельно приховають будь-які сліди чужинця, які за звичайних умов були б виявлені цими командами, що дозволяє зловмисникам підтримувати доступ до системи на рівні root'а (найвищий пріоритет) так, що системний адміністратор ніколи їх не побачить.

Наразі цей термін вже не обмежується UNIX-подібними ОС, оскільки інструменти зі схожим набором функцій існують і для не-Unix-подібних систем, таких як Microsoft Windows, незалежно від того, чи є в системі таке поняття як root.

Впродовж деякого часу в 2005 році Sony BMG використовувала програму захисту від нелегального копіювання CD-дисків, що автоматично встановлювалася на комп'ютери користувачів Windows. Sony BMG записувала на музичні диски програми Extended Copy Protection (XCP) та MediaMax CD-3. XCP записувався на 52 найменування, а MediaMax — на 50. Ці програми впливали на звичайний механізм роботи Microsoft Windows з музичними CD, відкриваючи діри в системі безпеки, якими могли скористатися віруси, та призводячи до інших проблем. Цей rootkit було визнано шкідливим, що змусило Sony BMG відкликати всі CD з цими програмами та випустити патч, що ремонтував завдані пошкодження.

(Див. http://blogs.technet.com/b/markrussinovich/archive/2005/10/31/sony-rootkits-and-digital-rights-management-gone-too-far.aspx)

Rootkit може взяти систему під свій повний контроль. Зазвичай задачею руткіта є приховування файлів, мережних з'єднань, адрес пам'яті або записів у реєстрі від інших програм, за допомогою яких адміністратори засікають навмисний чи ненавмисний привілейований доступ до ресурсів комп'ютера. Але rootkit може поєднуватися з іншиими файлами, задачі яких інші. Важливо зазначити, що, в той час як прив'язані до rootkit'а утиліти можуть біти шкідливими, сам по собі руткіт — всього лише технологія, яку можна використати як на шкоду, так і на благо.

Руткіти часто використовують для приховування утиліт. Це робиться з метою злісного використання доступу до системи, і часто полягає в утворенні "бекдорів", що дає змогу нападнику полегшити собі доступ до системи. Наприклад, руткіт може приховувати прикладну програму, що викликає оболонку при під'єднанні нападника до окремого мережного порта системи. Таку саму функціональність можуть мати руткіти рівня ядра. „Чорний хід" також може дозволити процесам непривілейованого користувача виконувати функції суперюзера. За допомогою руткітів можна приховати і всі інші види зловмисницьких інструментів, в тому числі засоби для організації подальших атак на комп'ютерні системи, з якими зв'язана вражена система, зокрема сніфери та кейлогери. Також заражений комп'ютер можна використовувати як плацдарм для подальшого використання («комп'ютери-зомбі»). Це робиться для того, щоб замаскувати справжнє джерело атаки. Засоби можуть включати механізми організації DOC-атак та розсилки спаму. Важливим прикладом застосування руткітів є засоби, що дозволяють автору руткіта бачити та доступатися до імен користувачів та реєстраційної інформації на відповідних сайтах. Автор руткіта може накопичувати індивідуальні набори персональної інформації з багатьох різних комп'ютерів. Це робить руткіти надзвичайно небезпечними, оскільки дозволяє різноманітним троянам використовувати приватну інформацію для доступу до особистої інформації..

Не завжди руткіти використовують для нападу та взяття комп'ютера під контроль. Деякі програми можуть використовувати руткіти, щоб сховатися від сторонніх сканерів, аби запобігти виявленню та втручанню в роботу. Відомо, що деякі програмні емулятори використовують ці методи. Alcohol 120% та Daemon Tools є прикладами комерційного використання нешкідливих рутітів..

Типи

Розрізняють 5 різновидів руткітів: руткіти рівня firmware, virtualized, kernel, library та application.

Firmware

Цей руткіт записується у вбудоване програмне забезпечення платформ. Це можливо, оскільки firmware далеко не завжди перевіряють на цілісність коду. John Heasman продемонстрував можливість створення таких руткітів в ACPI (Advanced Configuration and Power Interface) таблиці BIOS за допомогою AML (ACPI Machine Language) та в додатковій ROM PCI, AGP та PCIe пристроїв.

Віртуалізовані

Віртуалізовані руткіти — найнижчий з існуючих зараз рівнів. Ці руткіти працюють, змінюючи завантажувальну послідовність машини, аби замість звичних віртуальної машини чи операційної системи завантажилися вони. Після завантаження в пам'ять віртуалізований руткіт завантажує звичайну операційну систему в режимі віртуальної машини, що дозволяє руткіту перехоплювати абсолютно всі виклики гостьової ОС. Прикладом такого руткіту на базі віртуальної машини (VMBR) є руткіт „The **SubVirt**", розроблений Microsoft спільно з дослідниками з University of Michigan.

Рівень ядра

Ці руткіти додають свій код та/або заміщають ділянку коду ядра з метою приховання чорного ходу. Цього зазвичай досягають шляхом додавання до ядра нового коду через драйвер пристрою чи завантажувальний модуль, такий як Loadable Kernel Modules в Linux чи драйвери пристроїв у Microsoft Windows. Ці руткіти часто мають суттєвий вплив на стабільність системи в цілому, зокрема у випадку допущених при написанні такого руткіту помилок.

Руткіти рівня ядра можуть бути особливо небезпечними, оскільки їх важко виявити без відповідного ПО.

Рівень бібліотеки

Бібліотечні руткіти зазвичай доповнюють, перехоплюють чи заміщають собою системні виклики так, щоб виконувався код зловмисника.

Рівень застосування

Руткіти рівня застосування можуть заміщати двійковий код сравжніх програм троянізованими підробками або змінювати поведінку існуючих програм за допомогою хуків, патчів, ін'єкцій коду тощо.

Виявлення

Можливості будь-якої програми, що працює під підозрілою ОС, виявити руткіт, надзвичайно обмежені. Руткіти — це набір програм, що модифікують багато інструментів та бібліотек, від яких залежать всі програми в системі. Деякі руткіти змінюють запущене ядро через завантажувальні модулі (під Лінукс чи Юнікс) та VxD на платформах під Windows. Головною проблемою, пов'язаною з виявленням руткітів, є те, що запущеній ОС не можна довіряти. Іншими словами, не можна сподіватися, що такі дії, як запит списку всіх запущених процесів чи списку всіх файлів в каталозі, працюватимуть так, як планувалося. Детектори, запущені з-під вражених ОС, спрацьовують лише тому, що руткіти досі не навчилися повністю ховати себе.

Найкращим та найнадійнішим методом виявлення Р. є вимкнення комп'ютера та завантаження з альтернативного носія. Незапущений руткіт не може приховувати свою присутність, і найпопулярніші антивіруси виявлять руткіти за допомогою стандартних викликів ОС (перехоплюваних руткітом) та запитів низького рівня, що повинні бути надійними. Якщо є різниця, можна припустити наявність руткіта. Деякі руткіти намагаються захистити себе шляхом відстеження запущених процесів та призупинення своєї діяльності доти, доки сканування не завершиться.

Виробники захисних програм вбачають вирішення проблеми в інтеграції засобів виявлення руткітів в традиційні антивірусні продукти. Якщо руткіт вирішить сховатися в процесі сканування, його помітить стелс-детектор. Якщо він спробує тимчасово вивантажитися із системи, традиційний антивірус знайде його за відбитками діяльності. Така комбінована система захисту може змусити нападників впроваджувати в коди руткітів механізми контр-нападу (ретро процедури), які примусово вивантаживатимуть процеси захисного ПЗ з пам'яті, фактично вбиваючи антивірус. Як і у випадку з комп'ютерними вірусами, виявлення і знищення руткітів буде постійною боротьбою між програмістами по обидва боки барикад.

Існують кілька програм, здатних виявляти руткіти. На Юнікс-системах найпопулярнішими є chkrootkit та rkhunter. Для Windows-платформ існують багато безкоштовних програм, таких як F-Secure Blacklight. Іншим Windows-детектором є RootkitRevealer від Sysinternals, який виявить всі руткіти, порівнюючі дані, отримані від ОС, з лістингом, який він отримує власноруч. Але деякі руткіти стали додавати RootkitRevealer до списку файлів, від яких вони не ховаються. Через це відмінностей між двома списками не виникає, і детектор нічого не знаходить.

Як завжди, легше запобігти, ніж лікуватися. Якщо цілісність системних інсталяційних дисків не викликає сумнівів, для відстежування цілісності системи можна використовувати криптографію. Зберігаючи стан системних файлів одразу після свіжої інсталяції та після кожних змін системи, можна повідомляти користувача або адміністратора про будь-які потенційно небезпечні зміни системних файлів. При цьому використовується криптографічна хеш-функція, за допомогою якої отримують число фіксованої довжини, що залежить від значення кожного біту взятого файла. Регулярно порівнюючи ці хеш-функції, можна виявити зміни, що не робилися ані користувачем, ані системою.

Ще одним засобом боротьби з низькорівневими загрозами на кшталт руткітів є KMCS (Kernel Mode Code Signing) у Windows Vista. Система перевіряє, чи мають драйвери підписи з діючими

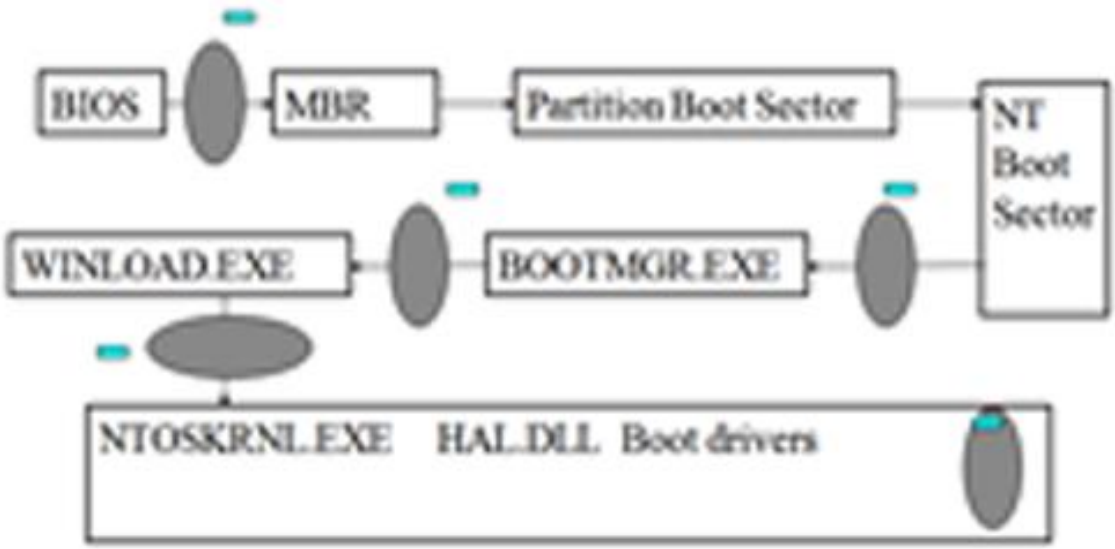
сертифікатами, більшість яких проходить через Windows Hardware Quality Lab (WHQL). Якщо підпису нема або вона фальшива, система розцінює драйвер як небезпечний і попереджає користувача, або взагалі відмовляється його встановлювати.

Видалення

Широко поширена думка про те, що це дуже непрактично. Навіть якщо відома природа та склад руткіта, час і зусилля адміністратора з відповідними навичками краще витратити на перевстановлення ОС з нуля. Оскільки програми клонування дисків (Symantec Ghost, Acronis True Image) роблять задачу відновлення чистої ОС майже тривіальною, не має потреби намагатися безпосередньо викорінювати руткіт.

Bootkits

Буткіти - руткіти з функцією завантаження з бут-секторів будь-яких пристроїв. В 2005 році на хакерській конференції Black Hat USA двоє дослідників з компанії eEye Digital Security - Derek Soeder та Ryan Permeh - представили концепт “BootRoot”. Технологія дозволяла розмістити в бут-секторі диска код, котрий потім перехоплював завантаження ядра Windows и запускав бекдор з можливістю віддаленого керування через локальну мережу. Ще за рік, на початку 2007, двоє індійських програмістів Nitin та Vipin Kumar представили Vbootkit – руткіт з функцією завантаження з бут-секторів будь-яких пристроїв, здатний працювати в Windows Vista.



(сірі місця – там, де буткіт перехоплює керування).

Незважаючи на те, що практично всі сучасні антивирусні програми мають функцію сканування завантажувальних секторів дисків, проблема виявлення перехоплених та змінених системних функцій досі не вирішена, навіть коли троянець і антивірус працюють в одній ОС, не кажучи вже про бекдор, що запускається раніше за операційну систему.

Історія з буткітом відображає весь спектр основних загроз інформаційної безпеки. У зв’язці з буткітами використовуються такі методи як зараження через браузер, руткіт-технології, ботнети, викрадення персональних даних, криптографія, обфускація, протидія антивірусним програмам.

Лекція 4.
Безпека програмного коду

— Програмне забезпечення як основне джерело проблем безпеки
— Шаблони атак на програмне забезпечення
— Відновлення початкового коду
— Основні підходи до зламу серверних застосувань
— Основні підходи до зламу клієнтських застосувань
— Злам за допомогою спеціально підібраних даних
— Переповнення буфера

Програмне забезпечення як основне джерело проблем безпеки

Є три основні проблеми, пов’язані з програмним забезпеченням, які перетворюють керування ризиком при використанні програм на основну задачу сучасності:

1) складність (сучасне ПЗ складається з мільйонів рядків коду)

2) розширюваність (переносимість коду сприяє поширенню вразливостей)

3) можливість взаємодії (зв’язок комп’ютерів в єдину мережу полегшує доступ до них)

Проблема захисту ПЗ – спеціалістам із забезпечення безпеки треба виявити усі вразливості, в той час як хакеру достатньо однієї.

Шаблони атак на програмне забезпечення

Структура шаблону атаки:
— вектор вторгнення
— проникнення в зону активації
— повідомлення про успіх
— зворотній зв’язок

Основні шаблони: впровадження власних команд, атака на базу даних, атака на привілейовану програму, DoS-атака на мережу чи окремий хост тощо.

Прийоми, якими користуються хакери:
— сканування мережі
— виявлення запущеної на комп’ютері-жертві ОС
— сканування портів
— визначення фізичної структури атакованої мережі
— пошук вразливих компонентів
— приховування свого місцезнаходження
— розміщення в системі таємних ходів для подальшого використання

Відновлення початкового коду

Інструменти відновлення початкового коду:
— відлагоджувачі (наприклад, SoftICE чи його „спадкоємець” Syser)
— засоби для внесення помилок (Hailstorm)
— дизасемблер (IDA)
— декомпілятор

Основні методи пошуку вразливих місць: „чорна скринька”, „біла скринька”, „сіра скринька”.

„Біла скринька” — аналіз передусім вихідного коду

„Чорна скринька” — дослідження за допомогою тестових даних

„Сіра скринька” — поєднання двох підходів (приклад — запуск програми в середовищі відлагоджувача і подання на вхід тестових даних)

Методи дослідження вихідного коду:
— відстеження обробки вхідних даних;
— використання відмінностей у різних версіях програми
— „Покриття коду” (code coverage)
— спроби доступу до драйверів ядра ОС
— перевірка на наявність витоку даних зі спільних буферів
— пошук недоліків в наданні прав доступу
— глобальний аудит

Основні підходи до зламу серверних застосувань

Проблема довіри до вхідних даних

Так звана „Безпека на боці клієнта”
Механізм „розширення привілеїв”
Під’єднання до вже запущеного процесу
Безпосередній доступ до командних файлів
Використання сценаріїв, вбудованих в інші сценарії

Методи атак на серверні застосування:
— впровадження команд для запуску командного інтерпретатора, готового приймати інструкції від віддаленого користувача
— використання каналів, портів і прав доступу
— використання властивостей файлової системи (використання косих, обернених косих та подвійних крапок для неконтрольованих мандрівок системою каталогів)
— маніпулювання змінними середовища
— використання зовнішніх змінних („приховані” змінні в PHP-сторінках)
— використання некоректної аутентифікації при відкритті сеансу
— підбір ідентифікаторів сеансу (підбір „ключа” до механізму генерування „псевдовипадкових” ідентифікаторів, аналіз фазового простору)
— використання додаткових можливостей аутентифікації
— використання некоректної обробки помилок

Злам клієнтських програм

Проблема довіри до даних і команд, надісланих сервером
Використання підмінених службових сигналів для атаки на клієнта (або сусідній сервер)
Переносимі сценарії (XSS – Cross-Site Scripting)
Завантаження і запуск шкідливого коду клієнтськими програмами (Word, Excel, Acrobat, Outlook тощо)
ActiveX і запуск неперевіреного коду
Атаки за допомогою шкідливого контенту або метаданих у «нормальних» файлах (mp3, pdf тощо)

Злам за допомогою спеціально підібраних даних

Техніка відстеження проходження даних у програмі:
— пошук ключових місць в коді (потенційно небезпечна ділянка, в якій обробляються дані користувача)
— трасування (під час виконання, зворотне від вразливого місця, пам’яті і буфера, «хід конем»)
— відновлення коду синтаксичного аналізатора (проблема косих рисок)

Переповнення буфера

Атаки, побудовані на помилках програмної реалізації, отримали велике розповсюдження, а їх інтенсивність з часом продовжує невпинно зростати. Величезна складність програмного забезпечення, часті випуски нових версій – все це призводить до погіршення якості програмного коду і недбалості його тестування. Більшість фірм, прагнучи привабити увагу споживачів, випускають на ринок ще сирі продукти і потім покращують їх вже в процесі експлуатації. Така схема створює гарне підґрунтя для злочинної діяльності, в якій використовуються помилки розробників.

Один із типів програмних помилок отримав назву «переповнення буфера» (*buffer overflow*). Якщо програміст виділяє буфер фіксованого розміру і заносить в нього динамічні дані, не переконавшись, чи достатньо вільного місця для їх розміщення, то дані, які не помістились в буфер, вилізуть за його межі і потраплять в комірки пам'яті, що знаходяться за кінцем буфера. Змінні в цих комірках виявляться деформованими, а поведінка програми стане непередбачуваною. Якщо буфер міститься у стеку, то існує можливість перезапису адреси повернення з функції, що призводить до передачі керування на незаплановану розробником ділянку коду.

Атаки такого роду можуть бути руйнівними, часто вони закінчуються отриманням повного контролю над комп'ютером. Теми переповнення та методів захисту від нього є дуже актуальними в наш час, адже навіть після десятиліть намагань повністю ліквідувати загрозу атак на переповнення помилки досі існують і успішно використовуються зловмисниками.

Переповнення буфера (buffer overflow) — явище, яке виникає, коли комп'ютерна програма записує дані за межами виділеного в пам'яті буфера. Переповнення буфера зазвичай виникає через неправильну роботу з даними, які були отримані ззовні, та пам'яттю за відсутності жорсткого захисту з боку підсистеми програмування (компілятора чи інтерпретатора) і операційної системи. В результаті переповнення можуть бути зіпсовані дані, які знаходяться за буфером (або перед ним).

Переповнення буфера є найпопулярнішим способом зламу комп'ютерних систем, так як більшість високорівневих мов використовують технологію стекового кадра – розміщення даних в стеку процесу і їх змішування з даними керування (такими, наприклад, як адреса початку стекового кадру, адреса повернення з виконуваної функції).

Іноді переповнення буфера навмисно використовується системними програмами для оминання обмежень в існуючих програмних чи програмно-апаратних засобах. Наприклад, операційна система iS-DOS (для комп'ютерів ZX Spectrum) використовує можливість переповнення буфера вбудованої TR-DOS для запуску свого завантажувача в машинних кодах (що неможливо зробити звичайними засобами TR-DOS).

Для кращого розуміння механізму переповнення наведемо аналогію. Якщо ви будете намагатись вкрасти що-небудь з магазину, то вам доведеться пробратись повз продавця. Продавець не стане творчо підходити до справи. Найвірогідніше, що він застосує тільки ті дії, які передбачені інструкцією. Інструкція – це великий вибір протоколів, які описують різні ситуації.

Приклад: "Контакт з водієм, який привозить товар":

- Крок 1. Взяти коробку
- Крок 2. Розписатись за коробку
- Крок 3. Переконатись, що водій від'їжджає.

Водій не може пройти повз продавця назад в магазин, оскільки в інструкції ясно сказано, що після отримання підписаної квитанції водій повинен поїхати.

Комп'ютери працюють схожим чином. Програми подібні до кроків в інструкції, комп'ютери виконують те, що написано в програмах і нічого більше. Атака може бути такою: прикинувшись водієм, можна поміняти сторінку інструкції продавця, коли він буде зайнятий підписом квитанції.

• Сторінка 165: Віддайте водію всі гроші з касового апарата. Див. наступну сторінку.

Це спрацює. Виконуючи інструкцію, продавець віддасть всі гроші з каси і подивиться наступну сторінку

• Сторінка 165: Запитайте у водія чи хоче він щось купити. Якщо хоче, див. сторінку 13, якщо ні, див. наступну сторінку.

• Сторінка 166: Попросіть водія поїхати.

Водію тільки залишається відповісти, що він не хоче нічого купити і поїхати. Можна використати такий спосіб обману, щоб переконати продавця пустити вас на склад або виконати якийсь інший план. По суті, це спосіб використання помилок переповнення буфера в комп'ютерних системах. Якщо комп'ютер запитує у користувача пароль, який має бути з 8 символів, а отримує натомість пароль з 200 символів, то додаткові символи можуть записатись в іншу область пам'яті. Якщо це підходяща область і в неї записати потрібні символи, то можна, наприклад, замінити команду «заборонити підключення» на «дозволити доступ» чи виконати ваш власний код.

Черв'як Морріса є найвідомішим прикладом використання помилки переповнення. Він використовує переповнення буфера в програмі для UNIX, яка мала ідентифікувати користувача за введеними даними. В ній не існувало обмеження на розмір введеної інформації, і введення більше ніж 512 байт призводило до переповнення буфера. Спеціальний довгий код Морріса дозволяв встановити його власну програму на комп'ютер і виконати її. Але і сам черв'як містив програмну помилку. Він мусив перестрибувати з комп'ютера на комп'ютер в Інтернеті, копіювати сам себе на кожен сервер і йти далі. Але помилка в коді призвела до того, що вірус копіювався необмежене число разів на кожному комп'ютері. Результатом був вихід заражених комп'ютерів з ладу. Крах відбувся на 6000 серверах Інтернету, на той час це було 10% від їх загального числа. Вміле програмування може передбачити атаки такого роду. Програма може скорочувати пароль до 8 символів, так що зайві символи не впишуться в пам'ять. Зробити це легко, але застосувати всюди важко. Проблема полягає в тому, що в будь-якій частині сучасного великого і складного коду є достатньо місць де можливе буферне переповнення (і які не настільки просто виявити, як у вищенаведеному прикладі). Дуже важко гарантувати, що немає ніяких проблем з переповненням, навіть якщо у вас був час для перевірки. Чим більший і складніший код, тим більша вірогідність атаки.

Найчастіше переповнення буфера зустрічається в програмах, написаних на C та C++. Зовсім легко переповнити буфер в асемблерній програмі, оскільки тут взагалі немає механізмів, які можуть запобігти переповненню.

В мовах ще вищого рівня програміст вже не має прямого доступу до пам'яті, хоч за це й доводиться розплачуватись продуктивністю. В такі мови, як Java, C# і Visual Basic, вже вбудовані рядковий тип, масиви з контролем виходу за межі і заборона на прямий доступ до пам'яті (в стандартному режимі). Хтось може сказати, що в таких мовах переповнення буфера неможливе, але правильніше було б сказати, що воно лише менш вірогідне. Адже в більшості ці мови реалізовані на C чи C++, а помилка в реалізації може стати причиною переповнення буфера. Ще одне потенційне джерело проблеми полягає в тому, що на певній стадії всі ці високорівневі мови повинні звертатися до операційної системи, а ось вона майже напевно написана на C чи C++. Мова C# дозволяє обійти стандартні механізми .NET, оголосивши небезпечну ділянку за допомогою ключового слова *unsafe*. Це спрощує взаємодію з операційною системою та бібліотеками, написаними на C/C++, але одночасно дає можливість припуститись звичайних для C/C++ помилок.

Процес виклику функції, передача параметрів і розміщення локальних змінних варіюється в різних мовах і залежить від конкретного компілятора, але в цілому виглядає приблизно так: в стек заносяться параметри, і значення регістра-вказівника стека зменшується, тобто стек росте від більших адрес до менших адрес.

Потім в стек заноситься адреса інструкції, яка йде за командою виклику підпрограми (в мікропроцесорах серії Intel 80x86 для цієї мети слугує інструкція CALL) і керування передається підпрограмі, яку викликають. Комірка пам'яті, в якій зберігається адреса

повернення, завжди доступна підпрограмі для модифікації. А локальні змінні (і буферні також) розташовуються компілятором в адресах, які знаходяться вище даної комірки.

Наприклад, стан стека при виклику функції myfunct() схематично можна зобразити так:

```
void myfunct()

{
        char a;
        char buff[5];
        char b;
...
}
```

Зміщення	Зміст комірок
0	A
1	buf[0]
2	buf[1]
3	buf[2]
4	buf[3]
5	buf[4]
6	B
7	Адреса повернення
8…	Стек функції, яка викликала myfunct

Спроба запису в комірку *buff[6]* призведе до викривлення адреси повернення, і після завершення роботи функції myfunct() відбудеться передача керування на зовсім незаплановану розробником коду ділянку. Все було б інакше, якби компілятор розташовував локальні змінні нижче комірки, в якій зберігається адреса повернення, але ця область стека вже зайнята, — вона належить функції, яка викликала myfunct(). Так влаштований стек: він зростає знизу вгору, а не навпаки.

1.1 Приклад переповнення буфера

Приклад, наведений нижче, наглядно ілюструє помилку програміста під назвою «зрив стека».

```
#include "stdafx.h"
#include <iostream>
#include <string.h>
using namespace std;

void    root()

        {
                printf("Hello, Root!\n");
        }
```

```
int     auth()
        {
                char user[10];
                char pass[10];
                printf("Login:"); gets(&user[0]);
                printf("Passw:"); gets(&pass[0]);
                if (!strcmp(&pass[0],"guest"))
                        return 1;
                return 0;
        }

int     main()
        {
        printf("Buffer Overflows Demo\n");
        if (auth())
                printf("Password ok\n");
        else
                printf("Invalid password\n");
        system("PAUSE");
    return 0;
        }
```

На перший погляд програма нібито мусить працювати нормально. Але функція *gets(),* яка читає рядок з клавіатури, не має ніякого уявлення про розмір виділеного під неї буфера і приймає дані доти, доки не зустріне символ повернення каретки. Якщо користувач введе в якості свого імені чи пароля стрічку, яка перевищує 10 символів, то її «хвіст» затре адресу повернення функції і подальше виконання програми виявиться неможливим.

Згідно «Нового словника хакера» Еріка Раймонда, помилки переповнення буфера це «те, що неуникно трапляється при намаганні запхати до буфера більше, ніж той може перетравити». Насправді це лише частковий випадок послідовного переповнення при записі. Крім цього існує індексне переповнення, яке полягає в доступі до довільної комірки пам'яті за кінцем буфера, де під «доступом» розуміється як операція читання, так і операція запису. Переповнення при записі призводить до затирання, а отже, спотворення однієї чи декількох змінних (включно з службовими змінними, такі, наприклад, як адреси повернення чи указники *this*), порушуючи цим нормальне виконання програми і призводячи до одного з можливих наслідків:

- нічого не відбувається;
- програма видає неправильні дані;
- програма «вилітає», зависає чи аварійно завершується з повідомленням про помилку;
- програма виконує незаплановані дії.

Переповнення при читанні менш небезпечне, так як «всього лише» призводить до потенційної можливості доступу до конфіденційних даних (наприклад, паролів чи ідентифікаторів TCP/IP з'єднання).

За кінцем буфера можуть знаходитись дані наступних типів: інші буфери, скалярні змінні і указники чи взагалі може не знаходитись нічого. Найбільшу загрозу безпеці системи становлять саме указники, оскільки вони дозволяють атакуючому здійснювати запис в довільні комірки пам'яті чи передавати керування за довільними адресами, наприклад, на початок буфера, що переповнюється, де розташований машинний код, спеціально підготовлений зловмисником, який зазвичай називають shell-кодом.

Буфери, що розташовані за кінцем буфера, що переповнюється, можуть містити деяку конфіденційну інформацію (наприклад, паролі). Розкриття чужих паролів чи нав'язування програмі свого пароля – типова поведінка атакуючого.

Скалярні змінні можуть зберігати індекси (і тоді вони фактично прирівнюються до указників), мітки, які визначають логіку поведінки програми та іншу інформацію.

Залежно від свого місця розташування, буфери поділяються на три незалежні категорії:

а) локальні буфери, які розташовані в стеці і які часто називають автоматичними змінними;

б) статичні буфери, які розташовані в секції (сегменті) даних;

в) динамічні буфери, які розташовані в купі.

Помилки переповнення – це фундаментальні помилки програмістів, які доволі важко відстежувати і фундаментальність яких забезпечується самою природою мови С – найпопулярнішої мови програмування.

Розглянемо функцію, яка визначає довжину переданого їй рядка, і яка посимвольно читає цей рядок до зустрічі з завершуючим його нулем. А якщо завершального нуля в кінці не виявиться? Тоді функція вийде за межі виділеного блоку пам'яті. В кращому випадку це закінчиться викиданням виключення. В гіршому – доступом до конфіденційних даних. Можна, звісно, передати максимальну довжину рядкового буфера з окремим аргументом, але чи справді вона буде правильною? Адже цей аргумент доводиться формувати вручну і він не застрахований від помилок. Отже, функції нічого не лишається, як покладатись на коректність переданих їй аргументів.

З іншого боку, виділення буфера можливе лише після обчислення довжини структури даних, що приймається, себто повинно виконуватись динамічно. Це запобігає розміщенню буферів у стеці, оскільки стекові буфери мають фіксований розмір, що задається ще на стадії компіляції. Зате стекові буфери автоматично звільняються при виході з функції, що запобігає потенційним проблемам з витоком пам'яті. Динамічні буфери, що виділяються з купи, набагато менш популярні, оскільки їх використання спотворює структуру програми. Якщо раніше обробка поточних помилок зводилась до негайного *return*'у, то тепер перед виходом з функції доводиться виконувати спеціальний код, який звільняє все, що програміст встиг до цього виділити. Без *goto* ця задача вирішується тільки глибоко вкладеними *if*-ами, обробниками структурних винятків, макросами чи зовнішніми функціями, що слугує джерелом багатьох і важко вловимих помилок.

Багато бібліотечних функцій (наприклад, *gets*, *sprintf*) не мають ніяких засобів обмеження довжини даних, що повертаються, і легко викликають помилки переповнення. Посібники з безпеки категорично забороняють використання таких функцій, рекомендуючи їх безпечні аналоги *fgets* і *snprintf*, які явно специфікують гранично допустиму довжину буфера, яка передається в спеціальному аргументі.

Крім невиправданого захаращення лістингу сторонніми аргументами і природніх проблем з їх синхронізацією (при роботі зі складними структурами даних, коли один-єдиний буфер зберігає багато всього, обчислення довжини «хвоста», що лишився, стає не такою вже й очевидною арифметичною задачею і тут легко припуститись грубих помилок), програміст стикається з необхідністю контролю цілісності даних. Як мінімум необхідно переконатись, що дані не було обрізано і, як максимум – коректно обробити ситуацію з обрізанням.

В C++ ситуація з переповненням набагато краща, хоч проблем все одно вистачає. Більшість реалізацій динамічних масивів працює повільно, а інакше й бути не може, адже побудова динамічних масивів змінної довжини – представлення їх вмісту у вигляді зв'язаної структури (наприклад, двонаправленого списку). Для швидкого доступу до довільного елементу список треба індексувати, а таблицю індексів десь зберігати. Таким чином, читання/запис одного символу потребує десятки машинних команд і багато звернень до

пам’яті (а пам’ять була і продовжує залишатись найвужчим місцем, яке суттєво знижує загальну продуктивність системи).

Навіть якщо компілятор контролюватиме межі масиву (одне додаткове звернення до пам’яті і три-чотири машинні команди), це не вирішить проблему, оскільки при виявленні переповнення відкомпільована програма не зможе зробити нічого розумнішого за аварійне завершення.

Якщо ж присутній виклик виключення і програміст забуде його обробити, то ми матимемо атаку типу «відмова в обслуговуванні”. Звісно, це не захоплення системи, але також недобре.

Отже помилки переповнення були і будуть.

Існує міф, що хакерам дуже легко переповнювати буфери і вони займаються цим постійно. Але насправді це далеко не легка задача. Ось неповний перелік обмежень, з якими доводиться стикатись черв’якам і хакерам:

- рядкові буфери не дозволяють впроваджувати символ нуля в середину буфера і перешкоджають введенню деяких символів, які програма інтерпретує особливим чином;
- розмір буферів, що переповнюються, виявляється катастрофічно малим для розміщення в них найпростішого завантажувача чи затирання значних структур даних;
- абсолютна адреса буфера, що переповнюються, атакуючому найчастіше невідома, тому доводиться оперувати відносними адресами, що дуже непросто з технічної точки зору;
- адреси системних і бібліотечних функцій змінюються від однієї операційної системи до іншої, ні на які адреси уразливої програми не можна покладатись, оскільки вони непостійні (особливо це актуально для UNIX-застосувань, що компілюються кожним користувачем самостійно);
- від атакуючого вимагається глибоке знання команд процесора, архітектури ОС, особливості різних компіляторів мов, багато часу для аналізу, проектування, відлагодження shell-коду.

Але існують і інші міфи – міфи захисників інформації, деякі з яких вважають, що можна захиститись від переповнення на 100 %. Насправді ж:

- не існує надійних методик автоматичного пошуку буферів, що переповнюються, які дають задовільний результат;
- всі розроблені методики боротьби з буферами, що переповнюються, знижують продуктивність, але не виключають можливості переповнення повністю, хоча й псують атакуючому життя;
- міжмережні екрани відсікають лише найпримітивніших черв‘яків, які завантажують свій хвіст через окреме TCP/IP з’єднання, відслідкувати передачу даних в контексті вже встановлених TCP/IP з’єднань міжмережний екран не в змозі.

Кінцева мета будь якої атаки – змусити систему здійснити щось «нехороше». Таке, що не можна здійснити легальним шляхом. Існує 4 способи реалізації атаки:

- читання секретних змінних;
- модифікація секретних змінних;
- передача керування на секретну функцію керування;
- передача керування на код, що передав жертві сам зловмисник.

1. Читання секретних змінних.

На роль секретних змінних в першу чергу претендують паролі на вхід в систему, а також паролі доступу до конфіденційної інформації. Всі вони розміщуються в просторі адрес

уразливого процесу, часто розташовуючись за фіксованими адресами (під «входом в систему» тут розуміється ім'я користувача і пароль, що забезпечує віддалене керування уразливою програмою).

Ще в просторі адрес процесу містяться дескриптори секретних файлів, сокети, ідентифікатори TCP/IP з'єднань і багато іншого. Вони можуть бути використані кодом, що передав жертві зловмисник і здійснити, наприклад, встановлення «невидимого» TCP/IP з'єднання, ховаючись за вже існуючим.

Комірки пам'яті, що зберігають вказівники на інші комірки, «секретними» не є, але знання їх вмісту значно полегшує атаку. Інакше атакуючому доведеться визначати опорні адреси навгад. Припустимо, в уразливій програмі міститься наступний код:

*char *ptr=malloc(MAX_BUF_SIZE);*

де ptr – вказівник на буфер, що містить секретний пароль. Припустимо також, що в програмі присутня помилка переповнення, що дозволяє зловмиснику читати вміст будь якої комірки простору адрес. Питання в тому, як цей буфер знайти. Сканувати все разом не тільки довго, але й небезпечно, так як можна наткнутись на невиділену сторінку пам'яті і тоді виконання процесу аварійно завершиться. Автоматичні і статичні змінні більш передбачувані. Тому атакуючий повинен спочатку прочитати вміст вказівника *ptr*, а потім вже секретний пароль, на який він вказує. Звісно, це лише приклад, яким можливості переповнюючого читання не обмежуються.

Саме ж переповнююче читання реалізується такими механізмами: «втратою» завершального нуля в рядкових буферах, модифікацією вказівників, індексним переповненням і нав'язуванням функції *printf* (і інших функцій форматованого виводу) зайвих специфікаторів.

2. Модифікація секретних змінних.

Можливість модифікації змінних дає значно більше можливостей для атаки дозволяючи:

- нав'язувати уразливій програмі «свої» паролі, дескриптори файлів, TCP/IP ідентифікатори тощо;
- модифікувати змінні, які керують розгалуженням програми;
- маніпулювати індексами та указниками, передаючи керування за довільною адресою (і також адресою, за якою знаходиться код, спеціально підготовлений зловмисником).

Найчастіше модифікація секретних змінних реалізується засобами послідовного переповнення буфера. Наприклад, якщо за кінцем буфера, що переповнюється, міститься указник на деяку змінну, в яку після переповнення щось пишеться, зловмисник може затерти будь яку комірку пам'яті (за виключенням комірок, явно захищених від модифікації).

3. Передача керування на секретну функцію програми.

Модифікація указників на виконуваний код призводить до можливості передачі керування на будь яку функцію уразливої програми (правда з передачею аргументів є певні проблеми). Практично кожна програма містить функції, що доступні тільки root'у, і які надають певні керівні можливості (наприклад, відкриття сесії, запуск файлів і т. д). В деяких випадках керування передається на середину функції (чи навіть на середину машинної інструкції) з таким розрахунком, щоб процесор виконав задум зловмисника, навіть якщо розробник програми не передбачав нічого подібного. Передача керування забезпечується або за рахунок зміни логіки виконання програми, або за рахунок підміни указників на код. І те, і інше спирається на модифікацію комірок програми.

4. Передача керування на код, що передав жертві зловмисник

Це різновид механізму передачі керування на секретну функцію програми, тільки тепер роль цієї функції виконує код, що підготував і певним чином передав зловмисник. З цією метою може використовуватись як сам буфер, що переповнюється, так і будь який інший буфер, доступний зловмиснику для безпосередньої модифікації і в момент передачі керування на shell-код, який присутній в адресному просторі уразливої програми (при цьому він повинен бути розташований за передбачуваними адресами, інакше передавати керування буде нікуди).

Методи захисту від переповнення буфера

1. Заміна небезпечних функцій роботи з рядками

Як мінімум потрібно замінити небезпечні функції типу strcpy, strcat і sprintf їх аналогами з лічильником. З нових розробок варто відмітити функцію strsafe, Safe CRT (бібліотеку часу виконання для C) і функції strlcat/strlcpy для *nix.

2. Слідкування за виділенням пам'яті

Ще одна причина переповнення буфера – арифметичні помилки. Потрібно уважно перевіряти всі місця, де для знаходження розміру буфера виконуються арифметичні обчислення.

3. Перевірка циклів і доступу до масивів

Переповнення можливо і у випадку, коли неправильно перевіряється умова виходу з циклу і не контролюється вихід за межі масиву перед записом в нього. Це одна з найважчих для знаходження помилок і вона часто проявляється не в тому модулі, де була допущена.

4. Захист стека

Захист стека вперше застосував Кріспін Коуєн в своїй програмі Stack-guard, Потім вона була незалежно реалізована Microsoft за допомогою прапорця компілятора /GS. В найпростішому варіанті суть захисту полягає в тому, що в стек між локальними значеннями і адресою повернення записується певне значення. Перед тим як процедура використає адресу повернення, ця величина перевіряється, щоб переконатися, що її не змінено. В більш сучасних реалізаціях для підвищення ефективності може також змінюватись порядок змінних. Перевага такого підходу в тому, що він легко реалізується, майже не знижує продуктивність і полегшує відлагодження у випадку псування стека. Інший приклад – програма ProPoliceStack, створена компанією IBM як розширення компілятора Gnu Compiler Collection (GCC). Будь який сучасний продукт має включати в себе захист стека.

5. Заборона на виконання в стеку чи купі (апаратний захист)

Цей контр-захід значно ускладнюю завдання хакера, але може відбитись на сумісності. Деякі додатки вважають допустимим виконувати код на льоту. Наприклад, це стосується мов Java і C#. Важливо також зазначити, що якщо супротивник зуміє змусити ваш додаток стати жертвою атаки з поверненням в libc, коли для досягнення певних цілей виконується виклик стандартної функції, то він зможе зняти захист зі сторінки пам'яті. Підтримка апаратного захисту залежить від процесора, операційної системи і навіть від її конкретної версії. Тому розраховувати на присутність в реальних умовах такого захисту не завжди варто, але все таки слід протестувати свою програму і переконатись, що вона буде працювати, коли стек і купа захищені від виконання. Для цього слід запустити її на тому процесорі і операційній системі, які підтримують апаратний захист. Наприклад, якщо

цільовою платформою є Windows XP, то потрібно зробити тести на машині з процесором AMD Athlon 64 FX під керуванням Windows XP SP2.

Апаратний **NX-Bit** (**No eXecute Bit** в процесорах AMD), чи **XD-Bit** (**Execute Disable Bit** в процессорах Intel), і пов'язана з ними технологія Microsoft **DEP** (Data Execution Prevention) – антивірусна технологія, яка перешкоджає зараженню комп'ютера деякими типами вірусів, які викликають переповнення буфера.

В архітектурі ARM схожа функціональність називалася XN (eXecute Never). Архітектура x86 мала схожу функціональність, починаючи з процесорів 80286, де вона була реалізована на рівні сегментів, але оскільки сучасні системи не використовують її у зв'язку з запровадженням моделі плоскої пам'яті (flat memory), то компанія AMD презентувала свою технологію NX-Bit, яка може контролювати виконання коду на окремій сторінці, а не на сегменті. Після широкого розповсюдження даної технології, компанія Intel додала схожу функціональність до процесорів Pentium 4.

Розглянемо, як дана технологія використовується різними операційними системами:

- MS Windows – Data Execution Prevention.
- Linux – підтримується на рівні ядра з 2004 року. Деякі дистрибутиви, такі як Fedora Core, Ubuntu і OpenSuse в своїх desktop-рішеннях надають ядро, скомпільоване без опції HIGHMEM64, яка необхідна для роботи з NX-бітом. Це зроблено для сумісності зі старими процесорами, які не підтримують NX-біт.
- Mac OS X - підтримує NX-біт на всіх Intel процесорах, починаючи з версії 10.4.4. В Mac OS X версії 10.4 NX-біт використовувався тільки для захисту стека. А починаючи з версії 10.5, присутній не тільки захист стеку, але і захист купи.
- OpenBSD - в даній системі технологія, яка реалізує захист від переповнення буфера називається W^X (вперше з'явилась в OpenBSD 3.3). Згідно даної технології сторінки пам'яті, до яких був доступ за замовченням, мали мітки невиконуваних.

Переповнення буфера в стеку

- черв'як Морріса використовував саме цей прийом, який був можливим через вразливість в протоколі Finger

(Цитата з оригіналу [2]:

"In the `fingerd` attack, it tries to infiltrate systems via a bug in `fingerd`, the `finger` daemon. Apparently this is where most of its success was (not in `sendmail`, as was originally reported). When `fingerd` is connected to, it reads its arguments from a pipe, but doesn't limit how much it reads. If it reads more than the internal 512-byte buffer allowed, it writes past the end of its stack. After the stack is a command to be executed ("`/usr/ucb/finger`") that actually does the work. On a VAX, the worm knew how much further from the stack it had to clobber to get to this command, which it replaced with the command "`/bin/sh`" (the Bourne shell). So instead of the `finger` command being executed, a shell was started with no arguments. Since this is run in the context of the `finger` daemon, `stdin` and `stdout` are connected to the network socket, and all the files were sucked over")

Переповнення буфера в купі

- приклад: вразливість в механізмі обробки JPEG-зображень

Причини:

— помилки з завершальним символом NULL у рядків;
— помилки при роботі з від'ємними числами
— помилки адресної арифметики