NVMe: New Vulnerabilities Made easy
Tal Lossos

Security Researcher
@CyberArk Labs

OS Internals :)
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>01</strong></td>
<td><strong>02</strong></td>
<td><strong>03</strong></td>
</tr>
<tr>
<td>Open-Source VR</td>
<td>SCA</td>
<td>SCA vs Kernel</td>
</tr>
<tr>
<td>Not Virtual Reality</td>
<td>Back to good ol’ SCA</td>
<td>Can it work?</td>
</tr>
<tr>
<td><strong>04</strong></td>
<td><strong>05</strong></td>
<td><strong>06</strong></td>
</tr>
<tr>
<td>Hunting for Bugs</td>
<td>NVMe Vulnerability</td>
<td>Demo</td>
</tr>
<tr>
<td>Yes it can :)</td>
<td>NVMe over TCP??</td>
<td>(plz work)</td>
</tr>
</tbody>
</table>
01

Open-Source VR

Methodologies & More
<table>
<thead>
<tr>
<th>Source code</th>
<th>Static analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easier to understand than by reversing</td>
<td>Write/Use tools for low-hanging-fruits</td>
</tr>
<tr>
<td><strong>Debugging</strong></td>
<td><strong>Re-compiling</strong></td>
</tr>
<tr>
<td>Run it &amp; debug on our machine!</td>
<td>Adding debug snippets</td>
</tr>
</tbody>
</table>
Open Source "disadvantages"

- **Bug Bounty**: Less (if at all) than closed source
- **Overwhelming**: So much code X.X
**STRATEGY**

**STEP 1**
Find code target

**STEP 2**
Static Analysis

**STEP 3**
Compile, Build & Execute

**STEP 4**
Repeat (& consider fuzzing)
Reverse Engineering

RE open-source??
Reverse Engineering

memcpy(to, from, size);
struct razer_report razer_chroma_extended_matrix_set_custom_frame2(
  unsigned char row_index, **unsigned char start_col, unsigned char stop_col**, unsigned char *rgb_data, size_t packetLength)
{
  const size_t row_length = (size_t) (((stop_col + 1) - start_col) * 3);
  const size_t data_length = (packetLength != 0) ? packetLength : row_length + 5;
  struct razer_report report = get_razer_report(0x0F, 0x03, data_length);
  // ...
  memcpy(&report.arguments[5], rgb_data, row_length);

  return report;
}
get_razer_report(&report, 0xFu, 3u, data_length);
report.transaction_id.id = 63;
report.arguments[2] = row_index;
if (row_length >= 8)
{
    *(QWORD *)&report.data_size + row_length) = *(QWORD*)((char *)l2_rgb_data + row_length - 8);
    qmemcpy(&report.arguments[10], (char *)l2_rgb_data + 5, 8 * ((row_length - 5) >> 3));
}
get_razer_report(&report, 0xFu, 3u, data_length);
report.transaction_id.id = 63;
report.arguments[2] = row_index;
if (row_length > 0x4D)
   return razer_chroma_extended_matrix_set_custom_frame2_cold();
if (row_length >= 8)
{
   *(report.data_size + row_length) = *(l2_rgb_data + row_length - 8);
   qmemcpy(&report.arguments[10], l2_rgb_data + 5, 8 * ((row_length - 5) >> 3));
}
“Dumb” Fuzzing?
Linux kernel communication

- Userspace
  - IOCTL
  - MMAP
  - Pseudo-FS
  - Netlink
- Read\Write
Linux kernel communication

- Userspace
  - IOCTL
    - Read/Write
  - MMAP
    - Pseudo-FS
    - Netlink
Linux kernel communication

Userspace

IOCTL

Read\Write

Pseudo-FS

Netlink

MMAP
Linux kernel communication

- Userspace
  - IOCTL
  - MMAP
    - Read\Write
    - Pseudo-FS
    - Netlink
Linux kernel communication

Userspace

IOCTL

Read\Write

Pseudo-FS

MMAP

Netlink
Example - CVE-2022-29021

```python
files = list(Path(path).rglob('matrix*'))

def fuzzer(max_length: int = 100, char_start: int = 32, char_range: int = 32) -> str:
    string_length = random.randrange(0, max_length + 1)
    out = ""
    for i in range(0, string_length):
        out += chr(random.randrange(char_start, char_start + char_range))
    return bytes(out, "utf-8")

while True:
    f = random.choice(files)
    i = fuzzer()
    print(f'trying file - {f}, input - {i}"
    with open(f, 'wb') as ff:
        ff.write(i)
```
02

SCA

Yes, Static code Analysis
“SCA commonly refers to the running of tools that attempt to highlight possible vulnerabilities within the source-code”

—OWASP
Available tools

C/CPP
Clang
cppcheck

Python
Pylint
Mypy

JS
ESLint
JSLint

CodeQL
SCA vs Kernel

The Linux kernel, ofc.
Memory Corruption Bugs

Buffer Overflow / Underflow  Integer Overflow / Underflow

Null PTR Dereference  Uninitialized variables
Cythoncheck

<table>
<thead>
<tr>
<th>OSS-Fuzz</th>
<th>Coverity Scan Build Status</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>oss-fuzz</td>
<td>coverity</td>
<td>license GPL3.0</td>
</tr>
</tbody>
</table>

About the name

The original name of this program was "C++check", but it was later changed to "Cythoncheck".

Despite the name, CPython is designed for both C and C++.
cppcheck

cppcheck source/XXX.c
cppcheck

```
./cppcheck source/XXX.c -xml 2>check.xml

htmlreport/cppcheck-htmlreport -
file=check.xml  -report-dir=check_report

firefox check_report/index.html
```
Cppcheck report

Defect summary

- 1549 integerOverflow
- 1390 unknownMacro
- 1356 uninitvar
- 59 syntaxError
- 54 nullPointer
- 50 nullPointerArithmetic
Possible null pointer dereference: ctrl->ctrl_key
ctrl->ctrl_key = nvme_auth_extract_key(host->dhchap_ctrl_secret + 10,
                                           host->dhchap_ctrl_key_key_hash);

if (IS_ERR(ctrl->ctrl_key)) {
    ret = PTR_ERR(ctrl->ctrl_key);
    ctrl->ctrl_key = NULL; // Assignment 'ctrl->ctrl_key=NULL', assigned value is 0
}

pr_debug("%s: using ctrl hash %s key %s ph\n", __func__,
          ctrl->ctrl_key->hash > 0 ? --- Null pointer dereference
          nvme_auth_hmac_name(ctrl->ctrl_key->hash) : "none",
          (int)ctrl->ctrl_key->len, ctrl->ctrl_key->key);
04

Hunting for Bugs

(using SCA)
SCA failed attempt

```
struct razer_report razer_chroma_extended_matrix_set_custom_frame2(
    unsigned char row_index, unsigned char start_col, unsigned char stop_col,
    unsigned char *rgb_data, size_t packetLength)
{
    const size_t row_length = (size_t) (((stop_col + 1) - start_col) * 3);
    const size_t data_length = (packetLength != 0) ? packetLength : row_length + 5;
    struct razer_report report = get_razer_report(0x0F, 0x03, data_length);
    // ...
    memcpy(&report.arguments[5], rgb_data, row_length);

    return report;
}
```
NVidia open-source drivers

[NVIDIA Linux Open GPU Kernel Module Source](https://github.com/nvidia/open-gpu-kernel-modules)

This is the source release of the NVIDIA Linux open GPU kernel modules, version 535.43.02.
Nvidia vulnerabilities

```c
void NV_API_CALL nv_acpi_methods_uninit(void)
{
    struct acpi_device *device = NULL;

#if defined(NV_ACPI_BUS_GET_DEVICE_PRESENT)
    acpi_bus_get_device(nvif_parent_gpu_handle, &device);
    nv_uninstall_notifier device->driver_data nv_acpi_event);
#endif
    device->driver_data = NULL;
    nvif_parent_gpu_handle = NULL;

    return;
}
```
```c
int acpi_bus_get_device(acpi_handle handle, struct acpi_device **device)
{
    return acpi_get_device_data(handle, device, NULL);
}

static int acpi_get_device_data(acpi_handle handle, struct acpi_device **device,
                                 void (*callback)(void *))
{
    acpi_status status;

    if (!device)
        return -EINVAL;

    *device = NULL;

    status = acpi_get_data_full(handle, acpi_scan_drop_device,
                                 (void **)device, callback);
    if (ACPI_FAILURE(status) || !*device) {
        ACPI_DEBUG_PRINT(( ACPI_DB_INFO,   "No context for object [%p]\n", handle));
        return -ENODEV;
    }
    return 0;
}
Linux kernel
As written about earlier this week, concerns have been raised over the "new" NTFS Linux driver that it's effectively unmaintained already less than one year after being mainlined. Linus Torvalds has since commented on the matter and opens up the door for other developers to maintain it.
```c
int attr_punch_hole(struct ntfs_inode *ni, u64 vbo, u64 bytes, u32 *frame_size)
{
    struct ATTRIB *attr = NULL, *attr_b;

    ...

    attr_b = ni_find_attr(ni, NULL, &le_b, ATTR_DATA, NULL, 0, NULL, &mi_b);
    if (!attr_b)
        return -ENOENT;

    if (!attr_b->non_res) {
        u32 data_size = le32_to_cpu(attr->res.data_size);
    }
```
```c
int attr_punch_hole(struct ntfs_inode *ni, u64 vbo, u64 bytes, u32 *frame_size)
{
    struct ATTRIB *attr = NULL, *attr_b;
    ...
    attr_b = ni_find_attr(ni, NULL, &le_b, ATTR_DATA, NULL, 0, NULL, &mi_b);
    if (!attr_b)
        return -ENOENT;
    if (!attr_b->non_res)
        u32 data_size = le32_to_cpu(attr->res.data_size);
```
nvme: In-band authentication support

From: Hannes Reinecke <hare-AT-suse.de>
To: Sagi Grimberg <sagi-AT-grimberg.me>
Subject: [PATCHv8 00/12] nvme: In-band authentication support
Date: Thu, 02 Dec 2021 16:23:46 +0100
Message-ID: <20211202152358.60116-1-hare@suse.de>
Cc: Christoph Hellwig <hch-AT-lst.de>, Keith Busch <keith.busch-AT-wdc.com>, linux-nvme-AT-lists.infradead.org, linux-crypto-AT-vger.kernel.org, Hannes Reinecke <hare-AT-suse.de>

Archive-link: Article
Possible null pointer dereference: ctrl->ctrl_key
ctrl->ctrl_key = nvme_auth_extract_key(host->dhchap_ctrl_secret + 10, host->dhchap_ctrl_key_hash);

if (IS_ERR(ctrl->ctrl_key)) {
    ret = PTR_ERR(ctrl->ctrl_key);
    ctrl->ctrl_key = NULL;
}

pr_debug("%s: using ctrl hash %s key % ph\n", __func__,
    ctrl->ctrl_key->hash > 0 ?
    nvme_auth_hmac_name(ctrl->ctrl_key->hash) : "none",
    (int)ctrl->ctrl_key->len, ctrl->ctrl_key->key);
ctrl->ctrl_key = nvme_auth_extract_key(host->dhchap_ctrl_secret + 10, host->dhchap_ctrl_key_hash);

if (IS_ERR(ctrl->ctrl_key)) {
    ret = PTR_ERR(ctrl->ctrl_key);
    ctrl->ctrl_key = NULL;
}

pr_debug("%s: using ctrl hash %s key %*ph\n", __func__,
    ctrl->ctrl_key->hash > 0 ?
    nvme_auth_hmac_name(ctrl->ctrl_key->hash) : "none",
    (int)ctrl->ctrl_key->len, ctrl->ctrl_key->key);
NVMe 101
NVMe

Nonvolatile Memory Express protocol, a transport protocol for accessing nonvolatile storage media over PCie.
NVMe-oF

Extension of NVMe which enables NVMe-based communication over connections other than PCIe, like FC or Ethernet (over fabric).
NVMe-TC

NVMe-TCP

Definition of NVMe-oF for TCP specifically.
NVMe - Protocol / Specification

NVMe-oF - Extension of NVMe to allow non-PCIe communication

NVMe-TCP - an implementation of NVMe-oF for TCP
NVMe-where?
Amazon EBS and NVMe on Linux instances

EBS volumes are exposed as NVMe block devices on instances built on the Nitro System. The device names are /dev/nvme0n1, /dev/nvme1n1, and so on. The device names that you specify in a block device mapping are renamed using NVMe device names (/dev/nvme[0-26]n1). The block device driver can assign NVMe device names in a different order than you specified for the volumes in the block device mapping.
Amazon EBS and NVMe on Linux instances

EBS volumes are exposed as NVMe block devices on instances built on the Nitro System. The device names are /dev/nvme0n1, /dev/nvme1n1, and so on. The device names that you specify in a block device mapping are renamed using NVMe device names (/dev/nvme[0-255]n1). The block device driver can assign NVMe device names in a different order than you specified for the volumes in the block device mapping.

Announcing the new Ebsv5 VM sizes offering 2X remote storage performance with NVMe-Public Preview
Amazon EBS and NVMe on Linux instances

EBS volumes are exposed as NVMe block devices on instances built on the Nitro System. The device names are /dev/nvme0n1, /dev/nvme1n1, and so on. The device names that you specify in a block device mapping are renamed using NVMe device names (/dev/nvme[0-26]n1). The block device driver can assign NVMe device names in a different order than you specified for the volumes in the block device mapping.

Announcing the new Ebsv5 VM sizes offering 2X remote storage performance with NVMe-Public Preview

NetApp NVMe solutions: Customer-focused technology leadership

Need help addressing your most stringent SLOs and business challenges? Look no further than our end-to-end NVMe-FC SAN solutions. They deliver the highest throughput and fastest response times yet for your enterprise workloads. And with new NVMe/TCP over traditional ethernet networks, the benefits keep rolling in.
Amazon EBS and NVMe on Linux instances

EBS volumes are exposed as NVMe block devices on instances built on the Nitro System. The device names are /dev/nvme0n1, /dev/nvme1n1, and so on. The device names that you specify in a block device mapping are renamed using NVMe device names (/dev/nvme[0–256]n1). The block device driver can assign NVMe device names in a different order than you specified for the volumes in the block device mapping.

Announcing the new Ebsv5 VM sizes offering 2X remote storage performance with NVMe-Public Preview

NetApp NVMe solutions: Customer-focused technology leadership

Need help addressing your most stringent SLOs and business challenges? Look no further than our end-to-end NVMe-FC SAN solutions. They deliver the highest throughput and fastest response times yet for your enterprise workloads. And with new NVMe/TCP over traditional ethernet networks, the benefits keep rolling in.

Linux Driver Information

The Linux NVMe™ driver is open source and included as part of the Linux Kernel, which can be found here https://github.com/torvalds/linux/tree/master/drivers/nvme

NVMe architecture works out of the box in every major operating system, including all mainstream Linux distributions. Please check on specific feature support with the distros, e.g. Red Hat Enterprise Linux, Ubuntu. NVMe technology has been supported since kernel 3.3, and at the time had been backported to 2.6. Intel released some history of the Linux NVMe drivers stack in 2015 here:
Exploitation
NULL PTR dereference

```c
int nvmet_setup_auth(struct nvmet_ctrl *ctrl)
{
    ctrl->ctrl_key = nvme_auth_extract_key(host->dhchap_ctrl_secret + 10,
                                            host->dhchap_ctrl_key_hash);
    if (IS_ERR(ctrl->ctrl_key)) {
        ret = PTR_ERR(ctrl->ctrl_key);
        ctrl->ctrl_key = NULL;
    }
    pr_debug("%s: using ctrl hash %s key %s\n", __func__,
             ctrl->ctrl_key->hash > 0 ?
                nvme_auth_hmac_name(ctrl->ctrl_key->hash) : "none",
             (int)ctrl->ctrl_key->len, ctrl->ctrl_key->key);
```
Invalid key

```c
key_len = base64_decode(secret, allocated_len, key->key);
if (key_len < 0) {
    pr_debug("base64 key decoding error %d\n", key_len);
    ret = key_len;
    goto out_free_secret;
}

if (key_len != 36 && key_len != 52 &&
    key_len != 68) {
    pr_err("Invalid key len %d\n", key_len);
    ret = -EINVAL;
    goto out_free_secret;
}
```
Exploiting NULL ptr dereference

- `mmap_min_addr` - min allowed address to map
- `SMEP/SMAP` - disable exec/access to userspace from kernel
- `panic_on_oops`
Project Zero

News and updates from the Project Zero team at Google

Thursday, January 19, 2023

Exploiting null-dereferences in the Linux kernel

Posted by Seth Jenkins, Project Zero
Hi all,

recent updates to the NVMe spec have added definitions for in-band authentication, and seeing that it provides some real benefit especially for NVMe-TCP here's an attempt to implement it.
ctrl->ctrl_key = nvme_auth_extract_key(host->dhchap_ctrl_secret + 10, 
host->dhchap_ctrl_key_hash);

if (IS_ERR(ctrl->ctrl_key)) {
    ret = PTR_ERR(ctrl->ctrl_key);
    ctrl->ctrl_key = NULL;
}

pr_debug("%s: using ctrl hash %s key %*ph\n", __func__,
         ctrl->ctrl_key->hash, 0 ?
         nvme_auth_hmac_name(ctrl->ctrl_key->hash) : "none",
         (int)ctrl->ctrl_key->len, ctrl->ctrl_key->key);
Call stack

\[ \text{nvmet\_setup\_auth}(\ldots) \]
\[ \rightarrow \text{nvmet\_execute\_admin\_connect}(\ldots) \]
\[ \rightarrow \text{nvmet\_parse\_connect\_command}(\ldots) \]
\[ \rightarrow \text{nvmet\_req\_init}(\ldots) \]
```c
bool nvmet_req_init(struct nvmet_req *req, struct nvmet_cq *cq,
                    struct nvmet_sq *sq, const struct nvmet_fabrics_ops *ops)
{
    u8 flags = req->cmd->common.flags;
    u16 status;

    req->cq = cq;
    req->sq = sq;
    req->ops = ops;
    req->sg = NULL;
    req->metadata_sg = NULL;
    req->sg_cnt = 0;
}
```

Referenced in 6 files:

- drivers/nvme/target/core.c, line 980
- drivers/nvme/target/fc.c, line 2549
- drivers/nvme/target/loop.c, line 151
- drivers/nvme/target/loop.c, line 184
- drivers/nvme/target/rdma.c, line 988
- drivers/nvme/target/tcp.c, line 999
- drivers/nvme/target/trace.h, line 61
NVMe-TCP

```c
static int nvmet_tcp_try_recv_pdu(struct nvmet_tcp_queue *queue)
{
    struct nvme_tcp_hdr *hdr = &queue->pdu.cmd.hdr;
    int len;
    struct kvec iov;
    struct msghdr msg = { .msg_flags = MSG_DONTWAIT };

    recv:
    iov.iov_base = (void *)&queue->pdu + queue->offset;
    iov.iov_len = queue->left;
    len = kernel_recvmsg(queue->sock, &msg, &iov, 1,
                         iov.iov_len, msg.msg_flags);
}```
NVMe Environment setup
setup --help

- NVMe / NVMe-TCP - Google (EASY)
setup --help

- NVMe / NVMe-TCP - Google (EASY)
- Authentication
setup --help

- NVMe / NVMe-TCP - Google (EASY)
- Authentication - Kernel Sources (HARD)
setup --help

- NVMe / NVMe-TCP - Google (EASY)
- Authentication - Kernel Sources (HARD)
- blktest framework :) (EASY)
blkttests

blkttests is a test framework for the Linux kernel block layer and storage stack. It is inspired by the xfstests filesystem testing framework. It was originally written by Omar Sandoval and announced in 2017.
function _create_nvmet_host() {
    local nvmet_subsystem="$1"
    local nvmet_hostnqn="$2"
    local nvmet_hostkey="$3"
    local nvmet_ctrlkey="$4"
    local cfs_path="${NVMET_CFS}/subsystems/${nvmet_subsystem}"
    local host_path="${NVMET_CFS}/hosts/${nvmet_hostnqn}"

    mkdir "${host_path}"
    echo 0 > "${cfs_path}/attr_allow_any_host"
    ln -s "${host_path}" "${cfs_path}/allowed_hosts/${nvmet_hostnqn}"

    if [[ "${nvmet_hostkey}" == "" ]]; then
        echo "${nvmet_hostkey}" > "${host_path}/dhchap_key"
    fi
    if [[ "${nvmet_ctrlkey}" == "" ]]; then
        echo "${nvmet_ctrlkey}" > "${host_path}/dhchap_ctrl_key"
    fi
}

Demo N.2

From Remote DoS to Pre-Auth Remote DoS
Authorization

[30749.638602] nvme nvme0: Connect for subsystem testnqn is not allowed, hostnqn: AAAA
Authorization

[30749.638602] nvme nvme0: Connect for subsystem testnqn is not allowed, hostnqn: AAAA

Frame 8: 1162 bytes on wire (9296 bits), 1162 bytes captured (9296 bits) on interface ens33, id 0
> Ethernet II, Src: VMware_a5:27:30, Dst: VMware_85:c4:5a
> Internet Protocol Version 4, Src: Dst:
> NVM Express Fabrics TCP, Fabrics Type: Connect (0x01) Cmd ID: 0x0000
  [Cmd Qid: 0 (AQ)]
  Pdu Type: CapsuleCommand (4)
>  Pdu Specific Flags: 0x00
>  Pdu Header Length: 72
>  Pdu Data Offset: 72
>  Packet Length: 1096
>  Cmd
>  Data
>
  Host Identifier: 8c124efaba94429d60672a438123d32
  Controller ID: 0xffff
  Reserved: 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
  Subsystem NQN: testnqn
    Host NQN: nqn.2014-08.org.nvmeexpress:uuuid:185f39a2-dc72-4991-8fe8-2f858de84678
  Reserved: 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
Fix
Reported the vulnerabilities which were patched

CVE
CVE-2023-0122
CVE-2022-4842
& more..

Blogpost

QR Code
Wrap-up

- Open source research might be a bit overwhelming - focus is the key
- If no documentation available, search for tests
- SCA is still a powerful tool for low-hanging-fruits!
THANKS!

@TalLossos