

翻译自：<https://www.zerodayinitiative.com/blog/2018/8/28/virtualbox-3d-acceleration-an-accelerated-attack-surface>

翻译人：Agostop

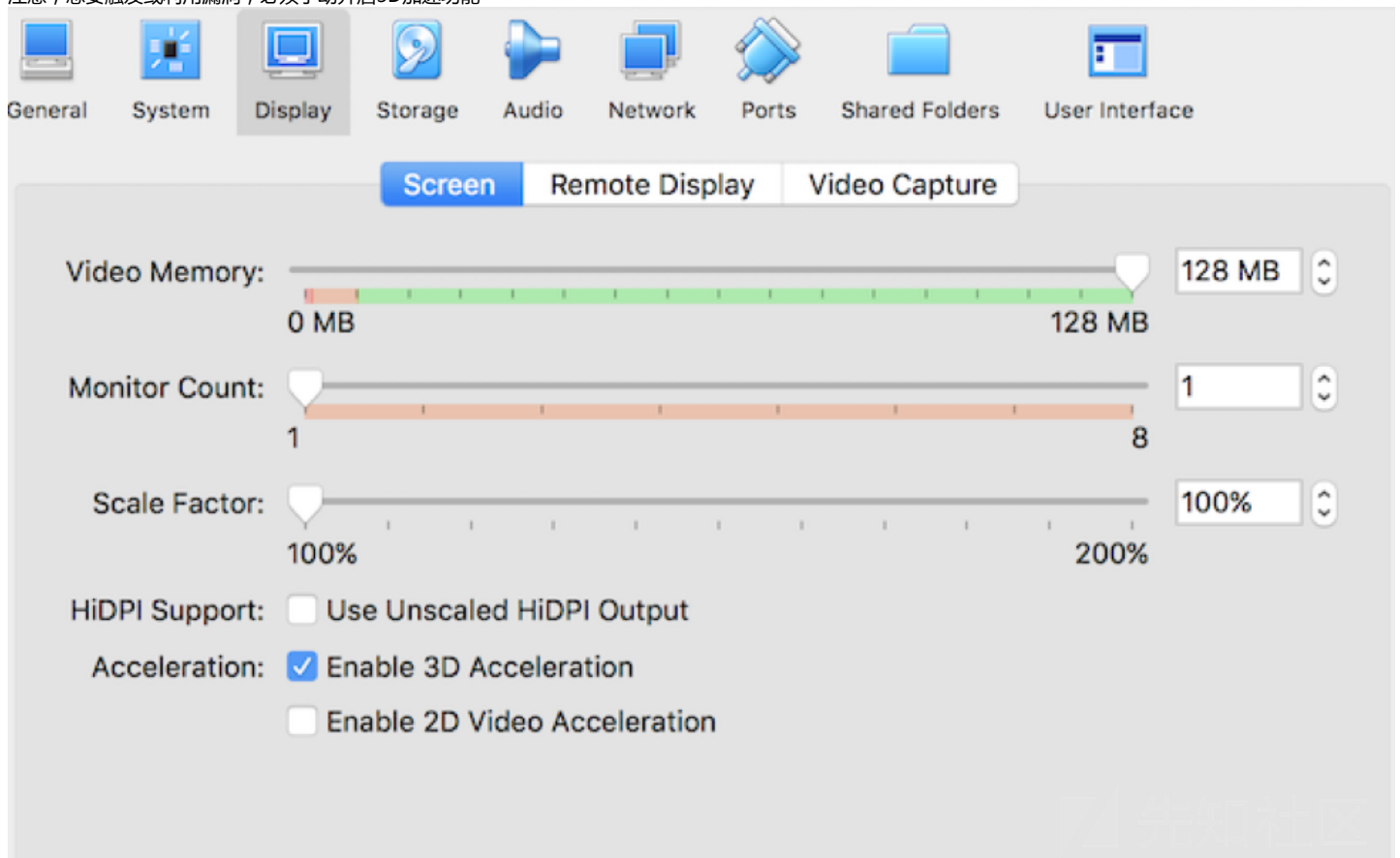
[VirtualBox](#) 是目前由Oracle Corporation开发和维持的一个免费的虚拟机管理程序。虽然可能鲜为人知，但它是VMware WorkStation和Microsoft Hyper-V的直接竞争对手。VirtualBox

3D加速功能使用[Chromium](#)库实现，我们从2017年7月开始接收它的漏洞。Oracle在他们的[文档](#)中警告不要启用3D加速功能，因为这个功能可能会带来安全隐患并且暴露受

在这篇博文中，我详细介绍了从2017年7月开始通过ZDI计划收到的不同类型的漏洞。我们收到的第一批漏洞是Vasily Vasiliev发现的，从那以后，这一领域的研究蓬勃发展，我们开始在这个受攻击面上接收更多的漏洞。

漏洞：

注意，想要触发或利用漏洞，必须手动开启3D加速功能



由于Chromium允许OpenGL图形渲染，所以当启用3D加速功能时，OpenGL应用程序会发送渲染相关命令给运行着Chromium服务器的管理程序。当开启这个选项的时候

接下来的讨论展示了影响OpenGL渲染命令的不同漏洞。

CVE-2018-2830：crUnpackExtendProgramParameters4fvNV中的整数溢出

这个经典的整数溢出漏洞存在crUnpackExtendProgramParameters4fvNV方法中，是由Vasily Vasiliev最早发现的，来看具体代码：

```
void crUnpackExtendProgramParameters4fvNV(void)
{
    GLenum target = READ_DATA(8, GLenum);
    GLuint index = READ_DATA(12, GLuint);
    GLuint num = READ_DATA(16, GLuint);
[1]   GLfloat *params = (GLfloat *) crAlloc(num * 4 * sizeof(GLfloat));
    if (params) {
[2]       GLuint i;
        for (i = 0; i < 4 * num; i++) {
            params[i] = READ_DATA(20 + i * 4, GLfloat);
        }
        cr_unpackDispatch.ProgramParameters4fvNV(target, index, num, params);
        crFree(params);
    }
}
```

先知社区

用户可以控制num变量，通过发送给num特定的值，可以在位置[1]发生整数溢出，导致params被分配很小的空间。之后在位置[2]，num被用在一个循环中，可以使变量params被写入内存。补丁

```
void crUnpackExtendProgramParameters4fvNV(void)
{
    GLenum target = READ_DATA(8, GLenum);
    GLuint index = READ_DATA(12, GLuint);
    GLuint num = READ_DATA(16, GLuint);
    GLfloat *params;

[1]   if (num >= UINT32_MAX / (4 * sizeof(GLfloat)))
    {
        crError("crUnpackExtendProgramParameters4fvNV: parameter 'num' is out of range");
        return;
    }

    params = (GLfloat *)crAlloc(num * 4 * sizeof(GLfloat));

    if (params) {
        GLuint i;
        for (i = 0; i < 4 * num; i++) {
            params[i] = READ_DATA(20 + i * sizeof(GLfloat), GLfloat);
        }
        cr_unpackDispatch.ProgramParameters4fvNV(target, index, num, params);
        crFree(params);
    }
}
```

先知社区

Oracle在[1]处添加了大小检查以避免溢出，这个漏洞已在[Oracle 4月份的补丁发布](#)中修复，并且被分配为CVE-2018-2830。有趣的是，这个漏洞同时被另外一个发现者Marcel Esage发现了一个具有相同攻击模式的非常相似的漏洞[ZDI-18-686](#)，在[7月补丁发布](#)中修复。

CVE-2018-2835：crStateTrackMatrixNV中的越界写

同样是由Vasily Vasiliev发现，但这个漏洞比前一个漏洞更严重，因为我们可以控制写入的地址和写入的值，漏洞的原理很简单：


```
void STATE_APIENTRY crStateTrackMatrixNV(GLenum target, GLuint address,
                                           GLenum matrix, GLenum transform)
{
    ...
    ...

    if (target == GL_VERTEX_PROGRAM_NV) {
        if (address & 0x3) {
            /* addr must be multiple of four */
            crStateError(__LINE__, __FILE__, GL_INVALID_VALUE,
                        "glTrackMatrixNV(address)");
            return;
        }

        switch (matrix) {
            case GL_NONE:
            case GL_MODELVIEW:
            case GL_PROJECTION:
            case GL_TEXTURE:
            case GL_COLOR:
            case GL_MODELVIEW_PROJECTION_NV:
            ...
            ...

            switch (transform) {
                case GL_IDENTITY_NV:
                case GL_INVERSE_NV:
                case GL_TRANSPOSE_NV:
                case GL_INVERSE_TRANSPOSE_NV:
                    /* OK, fallthrough */
                    break;
                default:
                    crStateError(__LINE__, __FILE__, GL_INVALID_ENUM,
                                "glTrackMatrixNV(transform = %x)", transform);
                    return;
            }
        }

    [1] p->TrackMatrix[address / 4] = matrix;
        p->TrackMatrixTransform[address / 4] = transform;
        DIRTY(pb->trackMatrix[address/4], g->neg_bitid);
        DIRTY(pb->dirty, g->neg_bitid);
    }
    ...
}
```



在[1]处，变量address用来作为TrackMatrix阵列中元素的索引，在这种情况下，address是用户可控的，所以可以被用来做越界写入。

补丁

```

void STATE_APIENTRY crStateTrackMatrixNV(GLenum target, GLuint address,
                                           GLenum matrix, GLenum transform)
{
    CRContext *g = GetCurrentContext();
    CRProgramState *p = &(g->program);
    CRStateBits *sb = GetCurrentBits();
    CRProgramBits *pb = &(sb->program);

    if (g->current.inBeginEnd) {
        crStateError(__LINE__, __FILE__, GL_INVALID_OPERATION,
                    "glGetTrackMatrixivNV called in Begin/End");
        return;
    }

    if (target == GL_VERTEX_PROGRAM_NV) {
[1]     if (address & 0x3 || address >= g->limits.maxVertexProgramEnvParams) {
        crStateError(__LINE__, __FILE__, GL_INVALID_VALUE,
                    "glTrackMatrixNV(address)");
        return;
    }
    ...
}

```



补丁在[1]处添加了额外的检查以确保地址不超过maxVertexProgramEnvParams。这个漏洞也在 [Oracle 4月份的补丁发布](#) 中修复，分配了CVE-2018-2835。

CVE-2018-2686 : crStatePixelMapuiv中基于堆栈的缓冲区溢出

欢迎回到90年代！这是在现代管理程序代码中你永远不会想到的一种基于堆栈的缓冲区溢出漏洞，是我最喜欢的漏洞类型之一，我们来看看代码：

```
void STATE_APIENTRY crStatePixelMapuiv (GLenum map, GLint mapsize, const GLuint * values)
{
    GLfloat fvalues[CR_MAX_PIXEL_MAP_TABLE];
    GLint i;

    if (!crStateIsBufferBound(GL_PIXEL_UNPACK_BUFFER_ARB))
    {
        if (map==GL_PIXEL_MAP_I_TO_I || map==GL_PIXEL_MAP_S_TO_S) {
[1]     for (i=0;i<mapsize;i++) {
            fvalues[i] = (GLfloat) values[i];
        }
        }
        else {
            for (i=0;i<mapsize;i++) {
                fvalues[i] = values[i] / 4294967295.0F;
            }
        }
        crStatePixelMapfv(map, mapsize, fvalues);
    }
    else
    {
        crStatePixelMapfv(map, mapsize, (const GLfloat*) values);
    }
}
```



用户控制了变量mapsize和数组values，之后在[1]位置，mapsize用于一个循环复制的操作中，最终导致了变量数组fvalues的溢出。

补丁

```
void STATE_APIENTRY crStatePixelMapuiv (GLenum map, GLint mapsize, const GLuint * values)
{
[1] if (mapsize < 0 || mapsize > CR_MAX_PIXEL_MAP_TABLE)
    {
        crError("crStatePixelMapuiv: parameter 'mapsize' is out of range");
        return;
    }

    if (!crStateIsBufferBound(GL_PIXEL_UNPACK_BUFFER_ARB))
    {
        GLfloat fvalues[CR_MAX_PIXEL_MAP_TABLE];
        GLint i;

        if (map==GL_PIXEL_MAP_I_TO_I || map==GL_PIXEL_MAP_S_TO_S) {
            for (i=0;i<mapsize;i++) {
                fvalues[i] = (GLfloat) values[i];
            }
        }

        ...
    }
}
```



Oracle通过在函数的最开头（[1]位置）添加了对mapsize的大小检查来修补此漏洞，以确保fvalues之后不会再溢出。
该漏洞在[Oracle的2018年1月修补版本](#)中进行了修复，并分配了CVE-2018-2686。

CVE-2018-2687 : crServerDispatchDeleteProgramsARB中的整数溢出

这个漏洞也是被Vasily Vasiliev所发现。糟糕的是，这种漏洞模式在代码中出现了很多次，来看看代码：

```
void SERVER_DISPATCH_APIENTRY crServerDispatchDeleteProgramsARB(GLsizei n, const GLuint * programs)
{
[1]   GLuint *pLocalProgs = (GLuint *) crAlloc(n * sizeof(GLuint));
      GLint i;
      if (!pLocalProgs) {
          crError("crServerDispatchDeleteProgramsARB: out of memory");
          return;
      }
[2]   for (i = 0; i < n; i++) {
      pLocalProgs[i] = crServerTranslateProgramID(programs[i]);
      }
      crStateDeleteProgramsARB(n, pLocalProgs);
      cr_server.head_spu->dispatch_table.DeleteProgramsARB(n, pLocalProgs);
      crFree(pLocalProgs);
}
```

由于变量n是用户可控的，可以在[1]处触发整数溢出，导致给变量pLocalprogs分配一个比较小的空间，之后在位置[2],n被用在一个复制循环中，从而可以使pLocalProgs也

补丁

```
void SERVER_DISPATCH_APIENTRY crServerDispatchDeleteProgramsARB(GLsizei n, const GLuint * programs)
{
    GLuint *pLocalProgs;
    GLint i;

[1] if (n >= UINT32_MAX / sizeof(GLuint))
    {
        crError("crServerDispatchDeleteProgramsARB: parameter 'n' is out of range");
        return;
    }

    pLocalProgs = (GLuint *)crAlloc(n * sizeof(GLuint));

    if (!pLocalProgs) {
        crError("crServerDispatchDeleteProgramsARB: out of memory");
        return;
    }
    for (i = 0; i < n; i++) {
        pLocalProgs[i] = crServerTranslateProgramID(programs[i]);
    }
    crStateDeleteProgramsARB(n, pLocalProgs);
    cr_server.head_spu->dispatch_table.DeleteProgramsARB(n, pLocalProgs);
    crFree(pLocalProgs);
}
```

和预期的一样，Oracle在[1]处添加了一个检查，以避免函数crAlloc中产生溢出。该漏洞在[Oracle的2018年1月修补版本](#)中进行了修补，并被分配了CVE-2018-2687。

这是能够触发这个漏洞的[PoC代码](#)，在Ubuntu访客VMs下使用。

总结

重用旧代码来渲染OpenGL图形并不是最好的选择，尤其是当需要在虚拟机管理程序中使用。凡事也要看到好的一面，如果使用者不考虑代码的过时和不安全性，对于代码

你可以看我的Twitter [@AbdHariri](#)，关注我们[团队](#)以了解最新的漏洞利用技术和安全补丁。

点击收藏 | 0 关注 | 1

[上一篇：java代码审计手书\(二\)](#) [下一篇：Windows版本QQ锁定可被绕过](#)

1. 3 条回复



[dotsu](#) 2018-11-26 18:59:43

djr说这个很垃圾

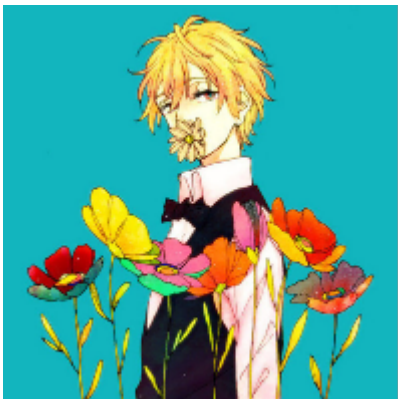
0 回复Ta



[Agostop](#) 2018-11-26 19:01:37

[@dotsu](#) 龙哥，你别被dj带偏啊

0 回复Ta



[一叶飘零](#) 2018-12-20 09:49:42

你好骚啊

0 回复Ta

[登录](#) 后跟帖

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

[目录](#)

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)