2021年十大漏洞利用

漏洞 (https://www.secrss.com/articles?tag=漏洞) · 网安国际 (https://www.secrss.com/articles?author=网安国际) · 2021-12-29

本文总结了作者心目中的2021年十大漏洞利用，重点考虑漏洞的影响范围和利用技术的创新度。

文/王铁磊

2021年即将过去。这似乎是一个没有什么存在感的年份。 这一年里，我们经历了2020欧洲杯，2020奥运会，整体感觉是把2020年重新过了一遍。但CVE编号还是不断提醒我，这是2021年了。而且，在漏洞攻击领域这是极为精彩的一年。

本文简单总结一下我心目中的2021年十大漏洞利用，重点考虑漏洞的影响范围和利用技术的创新度。因所知有限，我仅在我所能了解的领域做些总结，属于个人学习笔记。排名不代表先后。很多破解大赛上的精彩利用没能纳入表单，实属本人没来得及仔细分析。大家如果认同表单，说明我们有共识；如果不认同，说明技术世界很精彩，都是值得庆贺的事情。

01 iMessage 0Click 远程代码执行(CVE-2021-30860)

江湖上一直传闻NSO有iMessage 0Click远程代码执行的利用，但没人能想象是如何做到的。2019年，Google Project 0曾在iOS 12上利用 NSKeyedUnarchiver反序列漏洞重现过iMessage 的0Click攻击[1]。进入iOS 14时代后，Apple在iMessage数据处理链路上做了很多安全改进，包括引入重沙盒环境的BlastDoor服务处理不可信数据、不惜性能成本引入Dyld Shared Cache重新随机化机制等[2]。这使得iMessage 0Click攻击更加神龙见首不见尾。幸运的是，Project 0近期公开了针对iOS 14 iMessage攻击样本的分析报告，揭示了部分细节[3]。攻击是通过iMessage发送一个GIF图片实现的。而这个GIF图片，实际是一个畸形PDF文件。

漏洞发生在PDF文件解析过程，是一个平淡无奇的整数溢出导致的堆溢出。这个堆溢出的品相“看似”也一般，溢出时只能把对象指针越界写到临近内存。就是这么一个平淡无奇的漏洞，攻击者在JBIG2格式环境下构造了的精彩绝伦的利用过程。

• 首先，通过精细堆布局，使整数溢出转化为一个堆溢出，用JBIG2Bitmap对象指针覆盖原本的JBIG2Segment对象指针；此次覆盖非常重要，既能利用JBIG2Bitmap和JBIG2Segment的类型继承关系避免基于PAC(Pointer Authentication Code)的虚函数调用保护，又能及时截断溢出，避免过内存过度破坏造成崩溃。

• 溢出真正破坏的对象则是JBIG2Bitmap对象自身。JBIG2Bitmap对象中，使用32位整数表示Bitmap的长、宽等维度信息。溢出发生时，一个64位的JBIG2Bitmap指针，刚好会覆盖到JBIG2Bitmap对象的长、宽字段。iOS用户态指针基本是0x1xxxxxxxx形态，即高32位是1，低32位介于0x100000 and 0xffffffff之间。覆盖后，JBIG2Bitmap对象中的高度被指针的低32位覆盖，宽度被写成0x1。虽然不能精确控制高度，但是可以肯定的是，这个“高度”远超过JBIG2Bitmap本身的数据内存的长度。

• 后续使用这个被破坏JBIG2Bitmap时，就造成了一个越界写。通过这个受限的越界写，替换JBIG2Bitmap中的数据内存指针，就构造了一个真正的无边界Bitmap。

至此，我们已经可以领略到这个利用过程中堆布局的精细和巧妙。尽管如此，很多利用开发高手都可能实现这个任意读写能力。然而，虽然具备了任意读写，在无交互、无脚本执行的环境下，如何驱动这个任意读写能力，是制约漏洞利用的最大障碍。

这个利用接下来的操作就如同天外飞仙了。利用基于JBIG2 中的段命令实现与/或/非/异或等逻辑门操作，使用70,000多个段命令定义了自己的虚拟计算架构！该虚拟计算架构，有其虚拟寄存器，并且实现了64位的加法、比较操作；这意味着图灵完备的计算能力。

这个PDF解析是在IMTranscoderAgent进程中完成的。Project 0也将继续分享攻击如何进一步逃逸IMTranscoderAgent沙盒的，相信也十分精彩。还有一个很容易被忽视的细节，这个PDF解析的代码不是Apple自己开发的，而是使用了Xpdf的开源代码。那么问题来了，Xpdf里是否修复了这个漏洞？其他使用了Xpdf代码的产品是否修复了这个漏洞？对于其他受影响的产品，是否还需要如此复杂的攻击过程?

【参考资料】

1.https://googleprojectzero.blogspot.com/2020/01/remote-iphone-exploitation-part-1.html

2.https://googleprojectzero.blogspot.com/2021/01/a-look-at-imessage-in-ios-14.html

3.https://googleprojectzero.blogspot.com/2021/12/a-deep-dive-into-nso-zero-click.html

02 Apache Log4j2 远程代码执行(CVE-2021-44228)

时间拨回2016年。那一年的黑帽大会(Blackhat USA)上，来自Fortify的两个安全研究员Alvaro和Oleksandr做了题为《A Journey From JNDI/LDAP Manipulation to Remote Code Execution Dream Land》的报告[1]。报告详细介绍了JNDI/LDAP注入的原理、危害和攻击手段。时间再拨到2013年，有人向Log4j2开源社区提交支持JNDI Lookup 插件的请求并被采纳[2]。就这样，一个2013年就存在的洞，2016年就本应该被发现的问题，在2021年爆发了[3]。如果用一个词描述这个漏洞利用，我觉得“大道至简”比较合适。相比于漏洞利用，漏洞披露过程和场外声音更“嘈杂”，影响更广泛。

【参考资料】

1.https://www.blackhat.com/docs/us-16/materials/us-16-Munoz-A-Journey-From-JNDI-LDAP-Manipulation-To-RCE.pdf

2. https://issues.apache.org/jira/browse/LOG4J2-313

3. https://logging.apache.org/log4j/2.x/security.html

03 Zoom 0Click 远程代码执行(CVE-2021-30480)

毫无疑问，新冠疫情带来了Zoom的崛起。2021年4月份举办的Pwn2own破解大赛上，Zoom的远程0click项目奖金高达20万美金 [1]。Thijs Alkemade 和 Daan Keuper两人联手成功赢下了该项目并公开了技术细节[2]。

通常而言，通信类软件的0Click攻击难度系数大，主要是因为暴露给0Click的攻击面非常小。如何寻找和扩大攻击面是漏洞挖掘和利用开发的重点。而且由于现在操作系统平台上，普遍开启了地址随机化等防护机制，能否远程内存布局和泄漏从而绕过地址随机化成为重要障碍。

这次的0Click攻击，起源于一个堆溢出漏洞。攻击者仔细研究了Zoom通信协议后，掌握了远程堆布局的途径，然后利用Zoom的逻辑问题，直接和目标Zoom建立HTTPS通信；在TLS 握手过程中，利用堆溢出在目标Zoom内存中，覆盖URL请求里的0截止符(zero-terminator)；因为目标Zoom发送URL请求时，是根据0截止符判断URL的结束，而原本的0截止符被覆盖，导致目标Zoom会泄漏额外的非零数据，实现了远程的信息泄漏。在信息泄漏和远程堆布局的能力基础上，攻击者很容易再次溢出覆盖vtable或者函数指针等方式，利用ROP技巧实现任意代码执行。

回到堆溢出漏洞本身，漏洞根源在于，使用椭圆曲线的Diffie-Hellman密钥交换过程中，假定密钥长度不会超过1024；而一旦攻击者提供的密钥超过1024字节，就会造成堆溢出。无独有偶，Project Zero在Network Security Services (NSS) 密码库中也发现了一个内存溢出（CVE-2021-43527），原

因是NSS处理ASN.1 数字 signature字段时，默认签名不会超过2048字节[3]。NSS是一个被广泛使用的密码库。如果不是因为Log4j2的漏洞，现在安全圈可能还在排查CVE-2021-43527的影响范围[4]。再联想到心脏滴血漏洞，密码库代码实现问题依然严峻。

【参考资料】

1.https://www.zerodayinitiative.com/advisories/ZDI-21-971/

2. https://sector7.computest.nl/post/2021-08-zoom/

3.https://bugs.chromium.org/p/project-zero/issues/detail?id=2237

4.https://googleprojectzero.blogspot.com/2021/12/this-shouldnt-have-happened.html

04 Fugu14 iOS 完美越狱

有句戏言，“iOS完美越狱”比大熊猫还少。就这样，Fugu14 iOS 完美越狱发布了，还是开源的[1]。如果用一个词形容Fugu14里用到的漏洞，我会选择“优雅”。

• 首先来看用户态。为了加速App启动，iOS上启用了一个新的机制叫做“dyld closure”。App第一次启动后，系统会把这个App、dyld和动态库等资源链接好后存入一个名为.closure的缓存文件。这个Closure类似于App的运行时快照。App再次运行时，无需从新创建内存环境，而是加载这个Closure即可。但是这个closure文件的保存、加载和验证过程，都存在逻辑不足。Fugu14利用了这些特性，把普通App的Closure做了修改，将Closure中的主进程文件替换成了系统应用Spotlight，并修改了Closure的加载规则，从而获得了在Spotlight进程中的代码执行能力。Fugu14之所以选择Spotlight，是因为这个系统应用没有沙盒约束。

• 接下来看内核漏洞。在强大的用户态代码执行能力的基础上，Fugu14利用DriverKit的一个逻辑漏洞：创建一个IOUserClient，但是不去设置task成员；当task为0时，IOMemoryDescriptor::withAddressRanges直接操作物理内存地址。简单的说，Fugu14攻击内核时，直接获取了任意物理地址的读写能力。iOS 14实际上做了大量的针对虚拟内存管理的安全改进，但是这些面向虚拟地址的安全机制在任意物理地址读写能力面前全部失效。

• 简单的说一下PAC bypass。强如任意物理地址读写也并不能在iOS内核空间直接实现任意代码执行。基于ARM V8.3架构的PAC特性，iOS内核具备强大的控制流完整性保护。Fugu14在任意内存读写能力基础上，把内核正常接口thread_set_state转化成了一个签名gadget，从而达到劫持控制流的效果。这个PAC bypass技巧和我们的绕过技巧基本一致，下笔之时还在心痛。

• 最后看PPL绕过。这是整个攻击链条中最惊艳的一个环节。PPL是Apple在iOS上实现的一个基于硬件的物理内存页面保护机制[2]。简单的说，Apple基于定制化寄存器，确保只有PPL Text段代码才能修改特定物理页面。而PPL Text的代码，在修改特定物理页面之前，进行非常严格的检查。即使具备PAC bypass能力，可以在内核执行任意函数，也不能直接绕过PPL防护。过往PPL绕过技术也十分精彩[3]，但是很复杂。Fugu14中，有一个非常巧妙的新发现：对于一个 64位物理地址， PPL Text对这个地址进行权限属性检查时，会使用全64位值；然而一旦通过了PPL Text检查，硬件处理这个物理地址时，只使用低63位。借助这个软硬件逻辑不一致，Fugu14轻松的调用PPL Text段代码修改PPL保护页面。

Fugu14整个利用链条上，都是逻辑漏洞佐以内存破坏漏洞利用技巧。代码稳定，逻辑清晰，受益良多。

【参考资料】

1. https://github.com/LinusHenze/Fugu14

2. https://blog.siguza.net/APRR/

3. https://googleprojectzero.blogspot.com/2020/07/the-core-of-apple-is-ppl-breaking-xnu.html

05 iOS 15 远程越狱内核漏洞 (CVE-2021-30983)

写到这里，发现还没有自己的漏洞，必须发挥一下主场优势“夹带私货”了。这里简单介绍一下天府杯上iPhone 13 Pro远程越狱里的内核漏洞。

根据Apple的官方公告，漏洞发生在IOMobileFrameBuffer (IOMBF)内核扩展中[1]。IOMFB是iOS设备上一个非常重要的内核扩展，主要用于管理屏幕显示。诸如屏幕亮度、色调等参数的调节都是通过这个扩展实现的。熟悉iOS的同行可能注意到了，自从iPhone 12起，IOMFB大部分功能都已经移植到了DCP协处理器中。这个DCP协处理器运行单独固件，专门处理屏幕管理功能。

事实是，我们首先攻击了协处理器；在完全控制了DCP协处理器后，进一步攻击应用处理器。在iOS上，上一个通过协处理器攻击应用处理器的案例可能要追溯到2017年[2]。Project Zero在打掉iPhone上WiFi协处理器后，攻击了应用处理器。随着硬件升级，iPhone上装备了越来越多的协处理器，负责图像处理、运动感知等任务等。自2018年A12芯片起，iOS上引入了System Coprocessor Integrity Protection (SCIP)机制 [3]，强化协处理器运行环境安全。此次攻击为未来iOS漏洞研究提供了一个新的视角，把异构多核处理器核间通信的安全问题再次暴露出来。

【参考文献】

1. https://support.apple.com/en-us/HT212976

2. https://googleprojectzero.blogspot.com/2017/09/over-air-vol-2-pt-1-exploiting-wi-fi.html

3.https://support.apple.com/en-gb/guide/security/welcome/web

06 华为麒麟Bootrom漏洞

盘古实验室闻观行和Slipper在2021年7月30日MOSEC’21上披露了华为麒麟芯片Bootrom的一系列漏洞[1]；不到一周后，Blackhat USA上TASZK Security Labs对麒麟芯片可信启动链做了深入分析，也公开了一系列漏洞[3]。Bootrom是整个设备的可信基石，一旦Bootrom被攻破，设备再无安全可言。更为严重的是，由于Bootrom是固化在硬件中的代码，通常是无法通过软件升级进行漏洞修复。

历史上，iPhone手机的几次Bootrom漏洞洞都造成了长远影响。例如，2019年，苹果A5至A11系列芯片的Bootrom被发现存在UAF漏洞CVE-2019-8900，这个不能被修复的漏洞永久性地影响从iPhone 4S到iPhone X等型号手机[4][5]。

更令人吃惊的是，华为居然通过OTA升级，修复了Bootrom的漏洞！整个修复方案令人拍案叫绝。虽然华为没有公布修复细节，根据TASZK Security Labs的后续分析（对，他们最初也不理解为什么OTA升级能够修复BootRom的漏洞）[2]，OTA升级后会做一次额外的EFUSE write操作；正是这个write操作，完全禁止设备在启动阶段进入USB下载模式，从而避免了后续下载模式中的各种漏洞。

针对麒麟Bootrom漏洞，攻防双方联袂呈现了一场技术盛宴。一句话总结，攻的精彩，修的绝妙。

【参考资料】

1. https://github.com/hhj4ck/checkm30

2. https://labs.taszk.io/articles/post/huawei_kirin990_bootrom_patch/

3.https://www.blackhat.com/us-21/briefings/schedule/index.html#how-to-tame-your-unicorn---exploring-and-exploiting-zero-click-remote-interfaces-of-modern-huawei-smartphones-23337

4. https://github.com/axi0mX/ipwndfu

5. https://checkra.in/

07 Sudo 本地提权（CVE-2021-3156）

Sudo的堆溢出漏洞CVE-2021-3156获得了今年Pwnies最佳提权漏洞奖[1]。抛开这个漏洞的影响范围不谈，这个漏洞带给安全研究人员对*NIX环境下本地提权的经典重现就足够了：Suid-root 程序、命令行参数解析溢出、经典的触发模式、灵活的利用方法[2]，还有，漏洞已存在了10年之久。我们再欣赏一下这个POC:

$ sudoedit -s "\\" `perl -e "print "A" x 65536'`

除了上面引用的漏洞分析报告[2]，国外有个信息安全网红博主@LiveOverflow制作过一期视频，讲解了他对这个漏洞重现过程，尤其是分析了为什么这个漏洞没有被传统的Fuzzing工具挖掘出来，非常值得学习。Sudo这个漏洞，不仅影响Linux，也影响macOS；如果大家尝试在macOS上写这个漏洞的利用，就会更好的理解不同系统上漏洞缓解机制对漏洞利用开发的影响。换成M1 Mac更好玩。

【参考资料】

1. https://pwnies.com/winners/

2.https://www.qualys.com/2021/01/26/cve-2021-3156/baron-samedit-heap-based-overflow-sudo.txt

08 Windows Exchange Server ProxyLogon 远程代码执行

这是今年Pwnies最佳服务端漏洞大奖获得者[1]。Windows Exchange 是被广泛使用的邮件服务器。ProxyLogon[2]攻击允许攻击者在未授权的情况下，直接攻击Windows Exchange的443端口，获得服务器上远程命令执行。漏洞发现者Orange Tsai在Blackhat USA上分享了详细的漏洞细节[3]。

ProxyLogon掀开了攻击Windows Exchange的浪潮。一方面，Orange Tsai在2021年1月把ProxyLogon漏洞细节提交给微软；微软确认后计划于3月发布补丁。可是在补丁正式发布之前，2021年2月末，大量使用ProxyLogon漏洞的在野利用被捕获。在野利用和Orange Tsai提交给微软的POC一致。详细时间线可以参考[2]。

另一方面，Windows Exchange的各种攻击面相继曝光，各种新漏洞被发现。Orange Tsai在今年的Pwn2own大赛上使用另一套漏洞（ProxyShell）实现了Exchange的远程入侵；天府杯上，国内研究团队基于授权账户实现了Exchange远程代码执行(CVE-2021-42321)。

【参考资料】

1. https://pwnies.com/winners/

2. https://proxylogon.com/

3.https://i.blackhat.com/USA21/Wednesday-Handouts/us-21-ProxyLogon-Is-Just-The-Tip-Of-The-Iceberg-A-New-Attack-Surface-On-Microsoft-Exchange-Server.pdf

09 Windows PrintNightmare 远程代码执行漏洞

PrintNightmare 不是一个漏洞，而是一系列误会混合了一系列漏洞。漏洞发生在Windows系统的Print Spooler服务中，这是一个负责执行打印任务和打印机交互的服务。结合微软的公告和公开的技术分析[2,3,4]，我个人整理出来的时间线是这样的。微软将多个研究人员报告的漏洞，合并修复后发布本地提权漏洞 CVE-2021-1675的补丁；补丁修正的地方比较多，以至于安全研究员都以为漏洞被正确修复，于是公开了概念验证样本POC；然而很快被指出，CVE-2021-1675不仅没有修复好，而且还可能被远程触发。这样一来，一个不完备的补丁，暴露了远程可触发的安全漏洞。

噩梦就此开始。微软紧急发布CVE-2021-34527，进一步修复CVE-2021-1675残余的问题。可是故事还没有结束。CVE-2021-34527也没有完全解决问题，补丁仍然可能被绕过。微软又发布了CVE-2021-36958补丁。故事结束了吗？我不知道。我没有统计微软在2021年给Print Spooler发布多少补丁，据说现在打印服务的正常功能都已经受影响了。

如果大家还对漏洞细节感兴趣，可以参考 [1]。PrintNightmare这个案例再次证明，复杂软件中漏洞修复不是一个简单的事情。修编码错误容易，修机制性问题难于上青天。

【参考资料】

1.https://i.blackhat.com/USA21/Wednesday-Handouts/us-21-Diving-Into-Spooler-Discovering-Lpe-And-Rce-Vulnerabilities-In-Windows-Printer.pdf

2.https://www.rapid7.com/blog/post/2021/06/30/cve-2021-1675-printnightmare-patch-does-not-remediate-vulnerability/

3.https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-1675

4.https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-34527

5.https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-36958

10 微软MSHTML远程代码执行CVE-2021-40444

Log4shell爆发前，全球的安全公司可能正在围剿CVE-2021-40444 [1]。这是一个被在野利用的漏洞。攻击者通过恶意Office文档或者RTF格式的文档触发漏洞，实现远程代码执行。这个漏洞，投递方式丰富、利用简单、攻击稳定，注定是个长期风险。漏洞整个利用链条中，没有内存安全问题，而是组合了各种奇葩特性；在所有利用技巧中，从CAB文件解压释放INF文件中，存在路径穿越问题，可能是唯一的“明显”编码错误。

【参考资料】

1. https://www.microsoft.com/security/blog/2021/09/15/analyzing-attacks-that-exploit-the-mshtml-cve-2021-40444-vulnerability/

结语

至此已经写完了10个漏洞利用。其实还有很多精彩的漏洞利用没有涵盖，例如，天府杯上大宝同学的Chrome full chain攻击、Slipper的QEMU full chain攻击、@realBrightiup的iOS 15 Kernel Exp等。因为这些成果技术细节还没披露，我就不越俎代庖在文章里讨论了。文章准备过程中，咨询了很多朋友，这里就一并感谢！

漏洞 (https://www.secrss.com/articles?tag=漏洞)

相关资讯

Apache Struts代码执行漏洞 (CVE-2023-50164) 安全通告 (https://www.secrss.com/articles/61552)

漏洞 (https://www.secrss.com/articles?tag=漏洞) · 奇安信 CERT (https://www.secrss.com/articles?author=奇安信 CERT) · 8小时前

Apache OFBiz远程代码执行漏洞 (CVE-2023-49070) 安全通告 (https://www.secrss.com/articles/61534)

漏洞 (https://www.secrss.com/articles?tag=漏洞) · 奇安信 CERT (https://www.secrss.com/articles?author=奇安信 CERT) · 2023-12-07

Atlassian Confluence Data Center及Confluence Server远程代码执行漏洞安全风险通告 (https://www.secrss.com/articles/61463)

漏洞 (https://www.secrss.com/articles?tag=漏洞) · 奇安信 CERT (https://www.secrss.com/articles?author=奇安信 CERT) · 2023-12-06

评论（0）

登录后才能发表评论，请先 登录 / 注册 (https://www.secrss.com/login)