将挖洞当作爱好和职业的笑与泪：微软漏洞研究员的自白

原创

普通网友 于 2021-12-11 20:12:10 发布 192 收藏

文章标签： 信息安全 web安全 渗透测试 漏洞挖掘 安全

本文链接：https://blog.csdn.net/kali_Ma/article/details/121878394

前言

本文主人公目前是微软的一名漏洞研究员，他讲述了将挖洞当作爱好和当作职业的笑与泪，以及一些建议忠告。

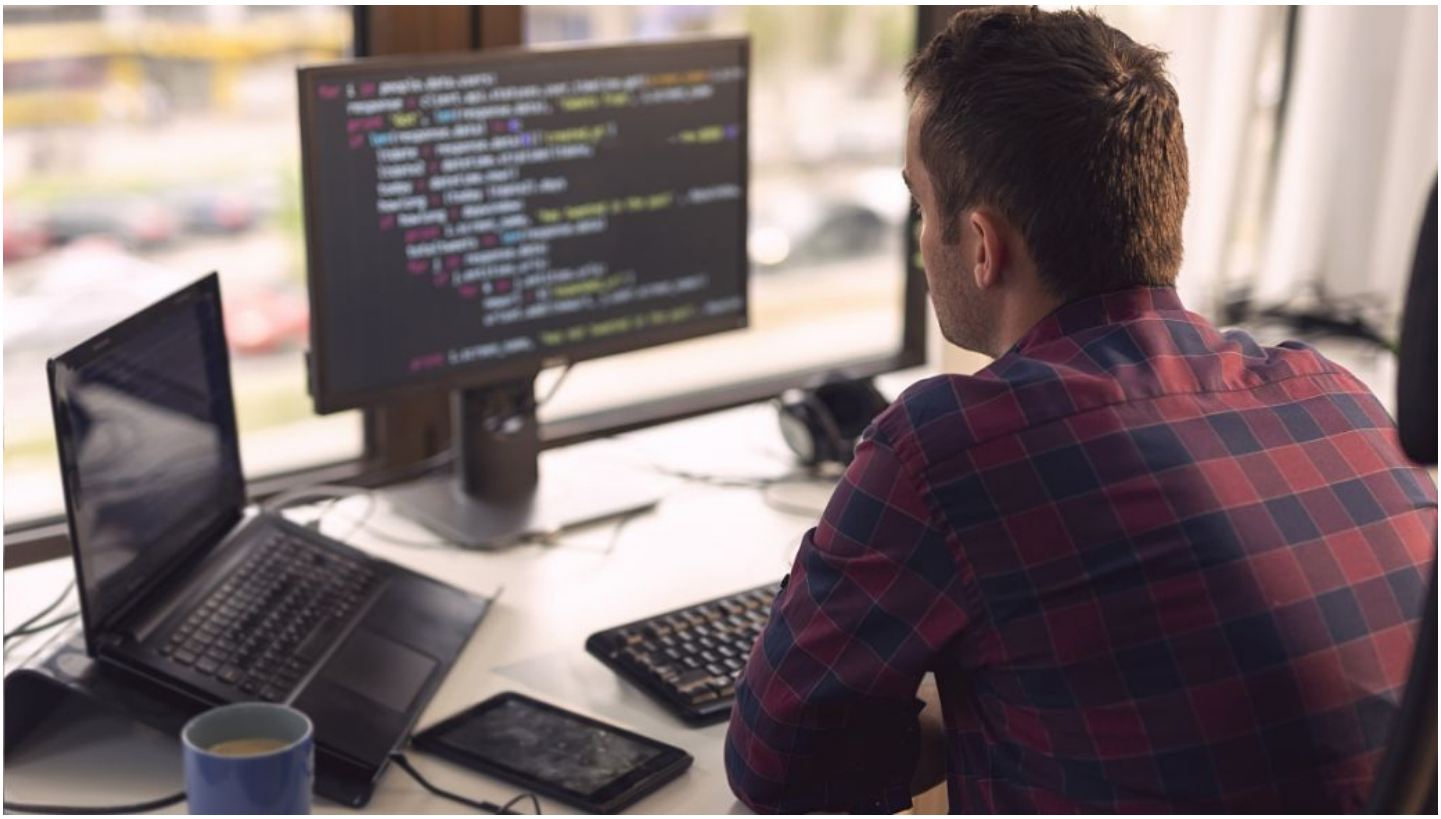
我写的通常都是自己挖到的有意思的浏览器 bug，希望能借读者一臂之力，应用这些经验或思路。不过这次我写的是将挖洞作为业余爱好与从事漏洞研究工作的不同之处。下面我将和你分享我的遗憾、惊喜和建议。虽然仅是一家之言，很可能无法代表你的观点。不过我仍然希望你能从中获得一些启发。

背景介绍

2015年年初，我获得了人生中的第一份官方漏洞奖励，也是从那时开始，我踏入了猎洞这一行。我挖的第一个漏洞是位于 O365 中的XSS 漏洞。之后，我在Firefox 挖到了第一个有效的浏览器漏洞。从此之后，我基本上是在查找浏览器漏洞。

2020年的后知后觉

我犯的最大一个错误是认为，由于自己几乎只专注于浏览器漏洞，对信息安全的热爱无法转换成安全职业。



从某种程度上讲，浏览器漏洞研究是一个细分领域，似乎我认识的多数漏洞猎人早期都是从 web 应用安全开始，而且通常会一直专注于这个领域，而我却对浏览器情有独钟并一头扎下去。这一事实让我的错误假设似乎成了板上钉钉的事。

但实际上正是因为浏览器安全领域的细分程度如此之高，所以符合主流浏览器厂商提出的浏览器安全职位要求的求职者并不多。如今我进入微软工作已超过一年，梦想照进现实，我们来聊聊本来可以如何更好地获得这个职位。

我从未想到自己会找到和浏览器安全相关的职位，于是忽视了企业为保护浏览器安全所做的一些事情。

我挖洞的主要方式是尽可能地阅读与浏览器及其安全相关的资料，之后手动测试自己的想法：虽然这种方法很耗时间，但我发现自己乐在其中，那种感觉和冥想差不多。大不了就当自己了解了一个新的Web API，但如果能找到一个有意思的安全漏洞就是双赢的结果了，而付出的成本不过是时间。

换句话说，我不仅从未真正进入 fuzzing 阶段，还对这种方法有了一种错误的看法，以为fuzzing就像是挖币，认为自己只需要运行一个 fuzzer 并滥用 CPU，等待有价值的崩溃发生即可。

我心中一直有个小人说这是浏览器安全的下一个阶段，但可能我掉入了固有方法的舒适区。

一言以蔽之：找到你的细分领域，就是让你兴奋而且能坚持下去的东西。做自己乐于做的事情你总会赢。

要想着修复漏洞

作为猎手，我所关心的就是找到有效的安全漏洞，提交，也可能还会为此进行讨论，之后继续去寻找下一个漏洞。这种做法让我的技能集出现了缺口，也错过了很多了解漏洞产生原理的学习机会。如果你也想尽快提交漏洞以免和别人重复，那么最起码应该花点时间研究下补丁。
我曾以为，“为啥我要免费给别人做事？”是吧？事实并非如此。这种思维为何会降低效率的原因如下：
1、了解了有问题部分的代码，假以时日你最终会了解到：比如 Chromium 的代码库以及其中的运行方式，它会让你挖洞越挖越顺；
2、 一旦发现了漏洞根因，你就可以查找这种存在漏洞代码的其它实例，最后，一个漏洞可能就变成了多个漏洞；
3、有了这种洞察力，你最终可能发现稍加修改，漏洞的严重程度就要比最初想的要更高；
4、你也可以付fei为 Chrome 等编写补丁，从而增加可获得的漏洞赏金。

在挖洞过程中，我做对的一件事就是参与漏洞报告和推特讨论。我阅读每个评论而且这些讨论中往往暗藏宝石。我也从来不遮掩对漏洞评估的反对意见，因为手握证明，我也会接收拒绝并愉快地继续前进。作为员工了解了后台的问题处理方式后，我发现很多事情都是合理的。之前不了解这一切时，我对结果是不接受的。
现在我的工作不仅要提交安全漏洞，而且还要和开发人员一起修复漏洞。
有时漏洞易于修复但并非一直如此。在提交漏洞时我必须额外注意，因为不仅要确保复现用例是最小化的，而且还要附上根因分析并给出修复建议。有时我会通过漏洞奖励计划提交漏洞，有时我会补充报告，便于开发人员能快速理解问题。
我们收到的大多数漏洞并非真正的漏洞。多数是重复提交的、设计问题以及无法复现或无效。很多时候漏洞描述并不直接，因此所有相关人员会一起讨论交流。这些会话中包含很多有意思的洞见，我总是很期待阅读这些邮件内容，如上所述，这些讨论内容中暗藏着宝石。
我经常会解释为何某些是漏洞某些不是，虽然这和此前的工作大不相同但仍然是类似的。因此，很明显之前报告的漏洞以及阅读评论的做法使我能够应付工作。

漏洞是不可避免的。随着Edge浏览器的发展，会出现新的特性、变更及改进，引入的代码自然会越来越多，产生的漏洞数量也会越来越多。更具体来说，漏洞是由如下方面引起的（适用于开发人员和安全团队）：

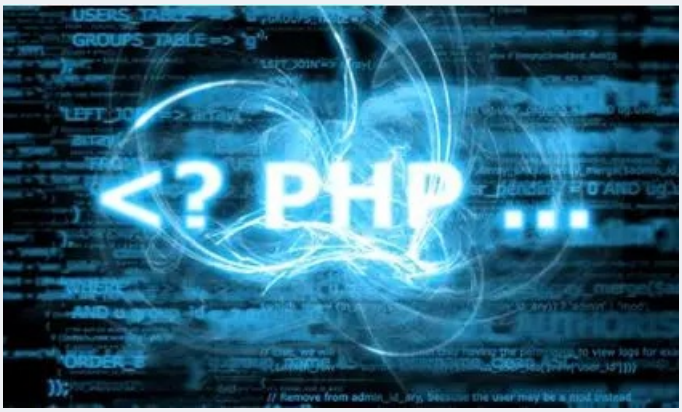
- **经验不足。**

没有人天生就理解庞大的 Edge 代码库的原理。理解需要时间，而犯错就是我们学习的方式。
不要假定最糟糕的情况。在保护安全方面，适度水平的焦虑是有帮助的；总是把假定的最糟糕情况作为潜在担忧并告知安全团队。

- **缺乏安全教育。**

我在计算机科学本科学习期间，学校并未开设安全课程。虽然课堂上可能会提到最佳编程实践，但它并非重点且远远不够，因此自学一些常见漏洞起着重要作用。
这份工作的一个重要部分是教授他人和自己学习安全知识。
安全漏洞的发现就是和所有相关人员进行讨论的机会，可以讨论漏洞如何运作以及未来如何通过自动化测试阻止它再次产生。安排强制性安全培训、记录最佳安全实践、以及每周发送安全小技巧邮件，让安全成为每个人的优先考虑因素。
早期我发现一件事：通常情况下，相比写出无懈可击的代码，找到安全漏洞更容易。因此，投资开发安全意识对于确保浏览器安全至关重要，但如果没有开发人员的协作，一切都是空谈。确保浏览器安全是所有团队的共同责任，而不应仅落在安全团队头上。
工作的另外一部分是对所有计划引入 Edge 的新特性进行安全审计。你可能认为这部分工作最像浏览器猎洞，那你可就错了。
尽管目标是找到特性中的 bug，但同时需要查看真正的 C++ 代码，而这是我在挖掘浏览器漏洞时很少做的事情。我必须马上学习一些 C++ 知识并粗略了解 Chromium-Edge 的代码库知识。虽然要花点时间但在同事的帮助下我hold住了。但实际上我希望自己之前应该更加关注漏

洞是如何修复的，而不是关注漏洞报告是否通过。
一言以蔽之：深入挖掘要比节省时间转向下一个漏洞重要得多。所以应该更加深入地挖掘并阅读代码。



自我怀疑

在整个旅程中，我常常会陷入自我怀疑。记得发现第一个漏洞之前我还在想，“我谁啊，要从这么大的公司找 bug？“但不管咋样，我还是傻傻地尝试了。当收到工作 offer 时，我想的是，”我谁啊，还能在这么大的公司工作？“不过我还是来了。
这种常见的感觉被称为“冒充者综合征”，要想克服它就得先了解它。忽视心中爱唱反调的小人人，试着做一些最初自己认为无法办到的事情，你就会被自己惊喜到。我在推特上比较活跃，仅关注处于安全圈子的人，并且坚持对圈子做贡献，这是我提升自信心的方式。
我讨论的并非是获取粉丝，而是获得自己所崇拜的牛人的鼓励让我一路向前。了解他人的工作鼓舞着我而且激励着我变得更好。成为团队的一分子也同样如此，和知识技能远在我之上的人一起工作让我变得更好，因为我周围都是很了不起的人。
一言以蔽之：勇敢点，去追求。成为支撑系统的一部分吧，不管它看起来是什么样子的。

分享就是关爱

大多数漏洞猎人都是自学成才，也就是从网上的免费资源学习。Writeup、博客、论文、幻灯片以及讨论安全的演讲和视频都是其他作者的努力。回馈并继续为这个开放资源做出贡献是自然而然之举，而且也会帮助他人实现目标。这样做带来的好处要比所花费的时间更有价值，比如：
1、如果你像我一样母语并非英语，那么通过上述活动可以锻炼自己的技术协作技能。有效沟通安全问题的能力，对于确保大家都理解并使工作顺利进行至关重要。
2、就某个话题进行写作能使自己重新了解这个领域，而且很有可能会获得之前错过的新东西。
3、这样做会让互联网更安全。你的贡献可能对其他人找到其它bug至关重要，因此会使更多的bug得到修复。
4、你会获得声誉和认可。需要招聘或协作的人员会因为你的写作而记住你。
当然，在社交媒体上你可能更加关注点赞数/阅读数/参与而非作品本身，我也会掉入这个陷阱。不要以点赞数的多少来衡量自己工作的价值，因为它是错误的。你可以把推特等社交媒体作为获益渠道，不过需要花一些时间关注对的人以及设置一些正确的词语。
我之前还有一种先入为主的想法，认为像微软这样的大企业不会关心在网上写东西的人。当我写下对某些公司的失望时，我只是在和同龄人发泄，而不会想到这些公司会有人关注。但事实上这些漏洞/推文对公司确实很重要，它们设置反馈功能并非为了好看，而真的是会有人看反馈。我的反馈得到认真对待，而设立目标是为了达到目标。这也是为何要写博客/推特/社交文章的另外一个原因，它真的会让你的声音被听到。
一言以蔽之：展现自己，让别人知道自己，让自己的声音被听到，你会发现自己身处一个超赞的社区。

掌握好自己的节奏

当我猎洞时，时间都是我的。我想挖的时候就去挖，如果感觉自己正被掏空，就会停下来做点别的。如果我有全职工作（进入微软之前），我就会放下安全研究工作做日常工作，所以从某种程度上讲我已经做到了平衡。
我在疫情期间加入微软，所以和很多人一样我也是在家办公。
作为新员工我想要学到所需学到的一切东西并尽最大努力做好这项工作，而在家办公也是我之前从未体验过的，它去除了那种人离开办公室并告诉大脑工作模式关闭的感觉。
这两个因素叠加到一起使我工作很长时间而且在周末也在工作。我那时特别兴奋，因此在控制节奏方面犯了一个大错。我忽视了团队成员给出的关于精力耗竭的建议以及微软给出的关于工作生活平衡的消息和提醒。
我感受到了这种倦怠。精神状态疲惫，感觉不适而且压力山大。于是我立即行动，停止在周末工作。我保证一天至少睡七八个小时并少喝咖啡。第一次度假时，我让自己不要想着工作的事情，回来后我的精神状态焕然一新。耗尽的感觉一扫而空，并做好了征服世界的准备。
一言以蔽之：掌握好自己的节奏，防止耗竭。这是一场马拉松而非冲刺。

如何进入浏览器安全领域？

这可能是我遇到的最常见问题之一。于我而言，我只是觉得浏览器漏洞很酷，这一点支撑着我去挖洞。除了之前提到的低自尊障碍外，答案就是这么简单。要知道，你自己必须要阅读很多东西，没有任何捷径，其实对于其它任何事情而言也一样，阅读自己感兴趣的一切至关重要，知识就是一切。
如下是一些可参考的入门：
1、**Chromium 代码库**——其中包含数百万行代码，查找你感兴趣的具体特性是如何运行的；Chromium bug tracker——某些被提交为安全漏洞的bug，其中包含深度讨论；Chromium security FAQ（安全常见问答）——涵盖了很多被发送为安全问题的误报；Chromium severity guide（严重性指南）——它会通过案例告诉你如何判定安全漏洞的严重程度。
2、**Mozilla bug tracker**——你可以找到影响 Firefox 6 的漏洞报告；Mozilla 安全公告——获取最新的 Firefox 安全漏洞修复方案，尽管通常而言，最近的几期中的漏洞尚未公开。
浏览器漏洞大概可以分为两种主要类型：
1、**内存相关漏洞**——释放后使用漏洞、缓冲区溢出及其朋友们，这些漏洞类型在 Chromium安全漏洞（高危及以上级别）中占比大概70%；
（1）进入 “about:crashes”，可以看到崩溃清单。对于Chrome 而言，可从如下位置找到崩溃转储：

```
C:\Users\test\AppData\Local\Google\Chrome\User Data\Crashpad\reports
```

（2）获得 winbg，打开这些崩溃转储并执行 !analyze –v
（3）通常通过fuzzing 就能找到这些漏洞。
2、 **逻辑漏洞**——全局XSS、SOP 绕过、UI欺骗等都是逻辑漏洞类型，和损坏内存没有任何关系。
（1）通常需要手动查找和/或阅读源代码
因为进入安全这一行之前，我基本从事的是web技术工作，因此有了一些 HTML/JS/CSS/HTTP的背景知识，所以会很自然地更加关注逻辑漏洞。因此，如果你已经具备一些 Web 知识，那专注于逻辑漏洞可能会更容易过渡一些。从另一方面来讲，如果你在逆向工程和/或自动化方面的经验更丰富，那就选择fuzzing 并重点关注内存损坏问题。当然，两个都抓也是可行的。
一旦大脑里面存储了足够多的信息，那么当你提出一些想法时就尝试去实践，尽管多数会失败，但要坚持不懈，最终你就会发现安全漏洞。
一言以蔽之：阅读、阅读、再阅读，要有创造性，失败之后多加尝试。

一切皆自动化

Fuzzing 在软件厂商保护自家应用程序过程中发挥着重要作用，自动化极具价值，因为它节约了很多时间并能获得很多成果。Fuzzing 就是将伪随机生成的输入投放到某个进程（如浏览器）中的过程，之后检测该进程是否崩溃。微软在fuzzer 中投入很多，很明显这是我必须补充的一个技能。
事实证明，我对fuzzer的误解是极其错误的。Fuzzing 就像是大海，本身就拥有很多种不同的方法论和学科。截至目前，我自己的经验只适用于浏览器上下文，而我只是刚刚接触而已，不过我已经感受到了它的巨大潜力。如下是我常用的一些工具/fuzzer/生成器，当然它们远非全部：
1、Libfuzzer：这是一个进程内覆盖率指南 fuzzer。以最基本的术语来解释就是，设置一个函数，便于将所有类型的输入抛进去并返回覆盖率信息（这里的“覆盖率”是指在既定输入中触及多少代码）。它要求能够访问浏览器源代码并具备一些 C++ 知识。
2、 Dharma：基于语法的fuzzer。由于这是同事 Christoph Diehl 创建的，我很高兴在遇到问题时可以去问他。你所需做的就是写一个语法文件，告诉它某个特定的 JavaScript API 或 HTML 元素是什么样的，它就会生成你可以在浏览器内运行的测试用例。它不需要具备浏览器源代码知识但要求具备 JS 和 HTML 知识以及某些关于语法文件的语法实践。
3、Playwright/Puppeteer —— 控制浏览器的 NodeJS API。这是一个很棒的工具，可以使你对浏览器具有很多控制。如果你需要使用浏览器执行一些重复任务，则会发现它是有用的。无需具备浏览器源代码背景，具有一些 NodeJS 背景即可。
4、Octo——Fuzzing 库。在fuzzing 过程中有时需要生成一个随机字符串或随机数等。这个库能使我们轻松访问还可用于绑定版本的 NodeJS 项目中或浏览器中。无需具备浏览器源代码背景，一些 NodeJS/JS 背景即可。
一言以蔽之：进入自动化；扩展自己的眼界，不要在舒适区待太久。
报告浏览器安全漏洞
安全漏洞将会一直存在，我认为这个事实在很长时间内都不会改变。就在你阅读本文时，就有一个bug等待你挖，所以赶紧去挖吧。
假设你在 Edge 中找到了一个漏洞，报告前建议先了解：
1、它能在Chrome 上复现吗？
（1）能——报告给 Chromium 而不是 Edge
（2）不能——报告给Edge
2、它能在Firefox 上复现吗？
（1）能——分别报给 Edge 和 Mozilla
（2）不能——那就只报给 Edge
3、你用的是Edge的最新稳定版吗？
（1）去 `“edge://settings/help”` 看看是不是最新的
（2）在报告中指出所使用的版本
4、在报告中指出所使用的操作系统（Windows、Mac、Linux、安卓或 iPhone）
5、在报告中上传一个最小的测试用例（不要发送实时演示链接）
6、上传一个关于该行为的短视频（我个人用的是OBS）

7、找到之前其它浏览器公开披露的漏洞作为先例
一言以蔽之：漏洞等待你去挖。

结论

希望这些经验会给你带来一些启发，你可以汲取其中的经验教训。其实我所提到的技巧只是流于表面，如果你遇到令人困惑的问题，就要深入挖掘了。搜索信息的能力同样是一项具有价值的技能，所以要尝试精进这种技能。如果卡壳了，遇到问题了，欢迎探讨。