剖析Linux病毒原型的工作过程和关键环节 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/252/2021_2022__E5_89_96_E6_9E_90Linu_c103_252889.htm 一、 介绍 写这篇文章的目的主要是对最近写的一个Linux病毒原型代码做一个总结，同时向对这方面有兴趣的朋友做一个简单的介绍。阅读这篇文章你需要一些知识，要对ELF有所了解、能够阅读一些嵌入了汇编的C代码、了解病毒的基本工作原理。 二、 ELF Infector (ELF文件感染器) 为了制作病毒文件，我们需要一个ELF文件感染器，用于制造第一个带毒文件。对于ELF文件感染技术，在Silvio Cesare的《Unix ELF PARASITES AND VIRUS》一文中已经有了一个非常好的分析、描述，在这方面我还没有发现可以对其进行补充的地方，因此在这里我把Silvio Cesare对ELF Infection过程的总结贴出来，以供参考： The final algorithm is using this information is.* Increase p_shoff by PAGE_SIZE in the ELF header* Patch the insertion code (parasite) to jump to the entry point(original)* Locate the text segment program header* Modify the entry point of the ELF header to point to the newcode (p_vaddr p_filesz)* Increase p_filesz by account for the new code (parasite)* Increase p_memsz to account for the new code (parasite)* For each phdr whos segment is after the insertion (text segment)* increase p_offset by PAGE_SIZE* For the last shdr in the text segment* increase sh_len by the parasite length* For each shdr whos section resides after the insertion* Increase sh_offset by PAGE_SIZE* Physically insert the new code (parasite) and pad to PAGE_SIZE, intothe file - text segment p_offset p_filesz (original)在Linux病毒原

型中所使用的gei - ELF Infector即是根据这个原理写的。在附录中你可以看到这个感染工具的源代码:
g-elf-infector.cg-elf-infector与病毒是独立开的，其只在制作第一个病毒文件时被使用。我简单介绍一下它的使用方法，g-elf-infector.c可以被用于任何希望--将二进制代码插入到指定文件的文本段，并在目标文件执行时首先被执行--的用途上。g-elf-infector.c的接口很简单，你只需要提供以下三个定义：* 存放你的二进制代码返回地址的地址，这里需要的是这个地址与代码起始地址的偏移，用于返回到目标程序的正常入口 #define PARACODE_RETADDR_ADDR_OFFSET 1232* 要插入的二进制代码（由于用C编写，所以这里需要以一个函数的方式提供）void parasite_code(void).* 二进制代码的结束（为了易用，这里用一个结尾函数来进行代码长度计算）void parasite_code_end(void).parasite_code_end应该是parasite_code函数后的第一个函数定义，通常应该如下表示 void parasite_code(void){.....}void parasite_code_end(void) {}在这里存在一个问题，就是编译有可能在编译时将parasite_code_end放在parasite_code地址的前面，这样会导致计算代码长度时失败，为了避免这个问题，你可以这样做 void parasite_code(void){.....}void parasite_code_end(void) {parasite_code().}有了这三个定义，g-elf-infector就能正确编译，编译后即可用来ELF文件感染 face=Verdana>三、病毒原型的工作过程 1 首先通过ELF Infector将病毒代码感染到一个ELF文件，这样就创造了第一个带毒文件，后续的传播就由它来完成。 2 当带毒文件被执行时，会首先跳到病毒代码开始执行。 3 病毒代码开始发作，在这个原型里，病毒会直接开始

传播。 4 病毒遍历当前目录下的每一个文件，如果是符合条件的ELF文件就开始感染。 5 病毒的感染过程和ELF Infector的过程类似，但由于工作环境的不同，代码的实现也是有较大区别的。 6 目前传染对ELF文件的基本要求是文本段要有剩余空间能够容纳病毒代码，如果无法满足，病毒会忽略此ELF。对于被感染过一次的ELF文件，文本段将不会有剩余的空间，因此二次感染是不会发生的。 7 病毒代码执行过后，会恢复堆栈和所有寄存器（这很重要），然后跳回到真正的可执行文件入口，开始正常的运行过程。 上面对病毒原型的工作过程的介绍也许显得千篇一律了，和我们早就熟知的关于病毒的一些介绍没有什么区别？是的，的确是这样，原理都是类似的，关键是要看实现。下面我们就将通过对一些技术问题的分析来了解具体的实现思路。 四、关键技术问题及处理 1 ELF文件执行流程重定向和代码插入 在ELF文件感染的问题上，ELF Infector与病毒传播时调用的infect_virus思路是一样的： * 定位到文本段，将病毒的代码接到文本段的尾部。这个过程的关键是要熟悉ELF文件的格式，将病毒代码复制到文本段尾部后，能够根据需要调整文本段长度改变所影响到的后续段(segment)或节(section)的虚拟地址。同时注意把新引入的文本段部分与一个.setion建立关联，防止strip这样的工具将插入的代码去除。还有一点就是要注意文本段增加长度的对齐问题，见ELF文档中的描述： p_align As ``Program Loading later in this part describes, loadable process segments must have congruent s for p_vaddr and p_offset, modulo the page size. * 通过过将ELF文件头中的入口地址修改为病毒代码地址来完成代码重定向： /* Modify the entry point of the ELF */ org_entry = ehdr->e_entry.

ehdr->e_entry = phdr[txt_index].p_vaddr phdr[txt_index].p_filesz. 2 病毒代码如何返回到真正的ELF文件入口 方法技巧应该很多，这里采用的方法是PUSH RET组合： __asm__ volatile (... "return:\n\t" "push $0xAABBCCDD\n\t" /* push ret_addr */ "ret\n" ::). 其中0xAABBCCDD处存放的是真正的程序入口地址，这个值在插入病毒代码时由感染程序来填写。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com