本文主要介绍一些 dotNet 加密保护工具的原理以及就其脱壳进行简单探讨。remotesoft protector、maxtocode、.Net Reactor、Cliprotector 、themida .Net、xenocode native compiler、DNGuard。

remotesoft protector

应该是一款比较老的。net 加密保护工具了，看其官方网站似乎还是 06 年更新过。该软件没有提供试用版下载， 相关资料比较少。去年接触过一个该软件保护的.Net 程序。加密后的程序发布时需要附带 native 的 dll。 这款壳可以算是 jit 层的壳，是 jit wrap 模式，通过 hook getJit 函数，拦截 jit 请求。在每次发生 jit 请求时其运行库会将加密的程序集完全“原地” 解密还原。

特点：整体解密
脱壳：拦截地层 jit 请求，然后中断。这时程序集已经完全解密，直接 pe dump 就行了。

maxtocode

这个大家应该比较熟悉了，和 remotesoft protector 应该时前后脚起步的关系吧。其 1.x，2.x，3.1x 和 3.2 内核有很大差别。
特点：单方法体解密

maxtocode 1.x 版本没有用过，不过 DST 组的菩提曾经写过 maxtocode 1.x 的脱壳机。

maxtocdoe 2.x 其内核是 EE 层，单方法体“原地”解密。编译之后再擦除解密的代码。

脱壳：因为是“原地”解密，所以方法体代码逃不过 profile 的。可以在 profile 里面记录每个方法体，然后填充到文件中。
方法二：nop 调 其内核 的擦除代码。这个不用修改其内核文件，只要还原 mscorwks。dll 中其 hook 的第二处地方即可。这样方法体解密后就在内存中了。所有方法 invoke 一面，直接 pe dump 即可。

maxtocode 3.1x，这个版本接触得比较多，我接触的第一个 maxtocode 版本就是 3.10。这一版其内核相对 2.x 变动比较大。方法体已经不是原地

解密的了，也就是说 profile 已经不能监视到其 il 代码了，这算是一个巨大的进步吧。3.1x 的内核基本上是一样的，只是后续的版本针对反射做了一些小动作。

脱壳：直接反射、修复后反射。
方法二：直接调用其内核的解密函数进行脱壳，简单快速。

maxtocode 2007 企业版，Jit 层内核 ，其在 ee 层和 jit 层均安装了多处 hook。其内核在前面的文章里面有详细介绍。

脱壳：因其 jit 层内核的漏洞，可以用简单的方式还原方法体。Hook Jit 后可以简单的进行

方法体还原完成单个方法的脱壳。
把每个方法都脱一面，填回文件即可。

.Net Reactor

一款很特别的。net 加密壳。它有两种模式， application 和 library。
第一种模式 是把.net 程序整体加密，然后创建一个 native 的 loader。整体加密的脱壳很简单，dump 内存即可。

第二种模式 加密后的程序集也要带一个 native 的 dll。和 maxtocode 一样，加了很多静态构造函数，一个 startup 函数。
但是在 startup 函数调用后，即完成了程序集的全部“原地”解密。所以运行后直接 dump 内存就可以了。

脱壳：直接 pe dump。

CliProtector

一款 jit 层的加密壳，大概是去年年底发现的。当时我在进行 DNGuard2.0 的开发，经分析后发现其内核模式和当时 DNGuard 2.0 的 jit 层内核很相似。分析后不久就发现了其 jit 层内核处理的一个漏洞，可以用简单的方式还原方法体。也就是最近在 maxtocode 2007 企业版中发现的那个。在我的 DNGuard 2.0 中对这个漏洞进行了预防处理。
个人感觉其模式兼容性比 maxtocode 2007 企业版要好。只是可惜，它除了有 jit 层漏洞，还偷了赖，IL 代码没有加密，和我出的 dnguard 1.0 demo 一样，只是把 il 搬了一下位置，没有加密。不过对于 jit 层脱壳来说加不加密倒无所谓了。但这样可能导致破解者从另一个角度去脱壳了。
特点：单方法体解密
脱壳：Jit hook，简单方法体还原， 同 maxtocode2007 企业版的脱壳方式。
方法二：分析其加密文件结构，直接还原（因其 il 代码没有加密，可以不用考虑解密算法的研究）。

themida .Net

themida 是 win32 的一个强壳，它支持 。Net 的加密，其加密方式是整体加密，但是凭借其 win32 anti 的优势，相比其它整体加密的加密工具来说强度要高一点，不过也就仅仅那么一点。

脱壳：过 anti，pe dump。

xenocode native compiler

xenocode 的专长是混淆保护，不过它也提供了一个所谓的生成本地代码的功能。其生成本地代码其实就是把 程序集打包，创建一个 native loader。但是它的打包把 framework 都包进去了，也就是说打包后的程序可以在没有安装 framework 的机器上直接运行，代价是生成的文件 体积非常大，因为它把十几兆的 framework 包进去了。

脱壳：直接 pe dump。
方法二：分析其打包的文件格式直接解包（已有工具）。

DNGuard 1.0 内核模式同 maxtocode 3.1x。脱壳方式也雷同。

DNGuard 2.0 Jit 层内核，同 maxtocode 2007 企业版和 CLIProtector。相比少了一个漏洞，不能用简单方式还原方法体。
如果破解者对 jit 内核工作非常熟悉，也能从 jit 层的结构体中重构出方法体。

脱壳：Jit hook 结构体重构模式。

总结：
以上除了 maxtocode 3.x, DNGuard, CLiProtector 外，其它工具加密的程序都存在 profile 漏洞，可以通过 profile 获取代码。

综合兼容性和强度 CLiProtector 和 maxtocode 2007 企业版 要好一些。
DNGuard 2.0 的强度好一些，兼容性比较差，就只支持 v2.0.50727.42 的 framework。

DNGuard 新版已经开始采用兼容全部 framework 的模式了。

上面的所有工具加密的程序集，都可以直接在 jit 层中截获 IL 字节码。 IL 字节码不是方法体，它是方法体的一部分。
只取得 il 字节码无法完成脱壳工作，但是已可反为 MSIL 汇编代码，进行算法分析了。

DNGuard HVM 的目标就是不让 jit 层截获可分析的 IL 字节码。