我们在天上的父，愿人都尊你的名为圣，愿你的国降临，愿你的旨意行在地上，如同行在天上，我们日用的饮食，今日赐给我们，免我们的债，如同我们免了人的债，不叫我们遇见试探，求我们脱离凶恶，因为国度，权柄，荣耀，全是你的，直到永远，啊们！！

This paper is 《Helping Johnny to Analyze Malware-A Usability-Optimized Decompiler and Malware Analysis User Study》

**一种可用性优化的反编译和恶意软件分析用户研究**

**1.** **Questions:**

1）文中提到的DREAM++，究竟在执行 6个具体的分析任务时，是如何提高分析师效能的 ？

2）如何从 安全工具可用性的角度，使得蓝盾的 安全工具，如 宏观态势感知，漏洞扫描，防火墙 等更易于操作和管理？

3）在 通用界面和原型设计上，又应该遵循怎样的原则？

Key Points from the video:

1. Expression Simplification
2. Decompilation type
3. Previous Work on Decompilation

• Do not focus on readability

• Do not include user studies in the evaluation

• Readability metrics:

• Compression ratio (smaller is better?)

• Number of gotos (less is better?)

1. Our Work on Decompilation

This work:

Usability extensions to DREAM

Malware analysis user study

1. Solved Readability Problems
2. User Study
3. Task Selection
4. Participants(参与者)
5. User Perception
6. Summary and Future Work

• Readability improvements to DREAM

• First malware analysis user study

• Human-centric approach can significantly improve the effectiveness of decompilers

• Focus on other use cases

• Vulnerability search in binary code

**2.摘要和解决思路**

Analysis of malicious software is an essential task in computer security, it provides the necessary understanding to devise effective countermeasures and mitigation strategies. The level of sophistication and complexity of current malware continues to evolve significantly, as the recently discovered "Regin" malware family strikingly illustrates. This complexity makes the already tedious and time-consuming task of manual malware reverse engineering even more difficult and challenging. Decompilation can accelerate this process by enabling analysts to reason about a high-level, more abstract from of binary code. While significant advances have been made, state-of-the-art decompilers still produce very complex and unreadable code and malware analysts still frequently go back to analyzing the assembly code.

In this paper, we present several semantics-preserving code transformations to make the decompiled code more readable, thus helping malware analysts understand and combat malware. We have implemented our optimizations as extensions to the academic decompiler DREAM. To evaluate our approach, we conducted the first user study to measure the quality of decompilers for malware analysis. Our study includes 6 analysis tasks based on real malware samples we obtained from independent malware experts. We evaluate three decompilers: the leading industry decompiler Hex-Rays, the state-of-the-art academic decompiler DREAM, and our usability-optimized decompiler DREAM++. The results show that our readability improvements had a significant effect on how well our participants could analyze the malware samples. DREAM++ outperforms both Hex-Rays and DREAM significantly. Using DREAM++ participants solved 3x more tasks than when using Hex-Rays and 2x more tasks than when using DREAM.

对恶意软件的分析是计算机安全的一项重要任务，它提供了必要的理解来进行设计和制定明确的战略。目前恶意软件的复杂程度和复杂程度在不断发展，就像最近发现的 "Regin" 恶意软件家族一样。这种复杂性使得手动恶意软件逆向工程的工作变得越来越麻烦和费时费力了。解析可以通过使分析人员从二进制代码的高层次，更抽象的角度来推动这一过程。（Decompilation can accelerate this process by enabling analysts to reason about a high-level, more abstract from of binary code.）。被制造，最先进的反编译器仍然产生非常复杂和不可读的代码，恶意软件仍然回到分析汇编代码。

在本文中，我们提出了几个语义保留代码转换（semantics-preserving code transformations），以使反编译代码，帮助恶意软件分析了解和打击恶意软件。提出的方法通过正确识别二进制函数[4，47]，可靠地提取二进制代码[7,35,37,42,58]加快分析过程，对模糊化的可执行代码进行模糊处理[20,56]，并恢复高级别从二进制代码抽象通过反编译

那么意软件分析人员仍然经常回到分析汇编代码。如何使反编译代码更可读？

**3.关于本文的解决方案：**

提出了几个语义保留代码转换（semantics-preserving code transformations），使反编译代码更易读，从而帮助恶意软件分析人员了解并打击恶意软件。

使用的优化方法分别为

表达式简化Expression Simplification

代码查询和转化Code Query and Transformation

控制流简化Control-Flow Simplification

语义感知命名Semantics-Aware Naming

————从五个方面来提高分析师效能

研究人员通过6个分析任务去对3款翻译器工具进行评价，她们是：

* 工业界领先的Hex Rays
* 学术界领先的DREAM
* 本研究组的增强版反编译工具DREAM++

研究结果表明，使用DREAM++能使 安全分析人员的 反编译效能有 数倍的提升

————从6个分析任务和用户来研究测试以证明反汇编的优化与可读性

**反汇编工具的可用性检测及结果：在文中**研**究人员通过6个分析任务去对3款翻译器工具进行评价**

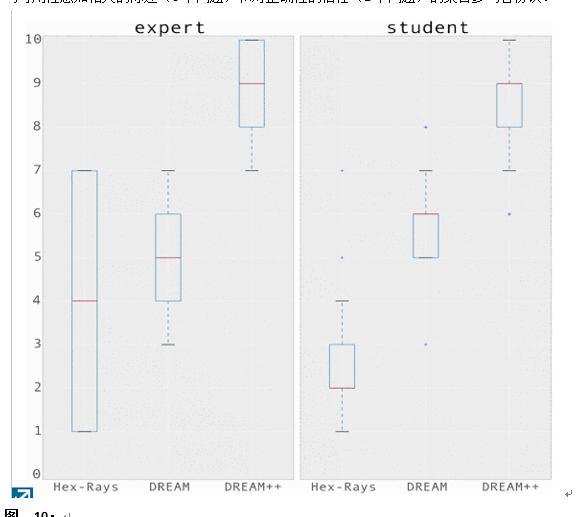
我们向我们大学完成了恶意软件新手训练的学生发送了36封邀请函

我们还邀请了来自商业安全公司的31名恶意软件专家

然后经过多次筛选，7名参与者报告恶意软件分析经验不到一年，一名参与者报告说有14年的恶意软件分析经验，17名恶意软件分析师开始研究，测试

经过多次测试，如下图可以明显的看出：可以清楚地看到，Dream ++竞争得分高，

——————由此可看出Dream ++是一个容易上手且功能强大的反编译工具



思考：

1）反编译技术在恶意软件检测中的实际应用有哪些？

2）是否可以将反编译的技术融入到蓝盾的安全全产品当中，以增强蓝盾产品的效能。

3）针对于反编译技术在计算机安全方面的发展，是否会有新的突破及扩展？

主要内容

1.问题陈述和概述

**1）复数表达式**

最先进的反应堆通常产生过于复杂的表达。这样的表达式很少在人类编写的源代码中找到，因此很难理解。这包括

**a）复杂逻辑表达式Complex Logic Expressions**

**b）变量数量Number of Variables**

**c） 指针表达式 Pointer Expressions**

**2）卷积控制流程**

程序的可读性在很大程度上取决于其排序控制的简单性[23]。两个问题往往使反编译器回收的控制流程复杂化

**a）重复/内联代码Duplicate/Inlined Code**

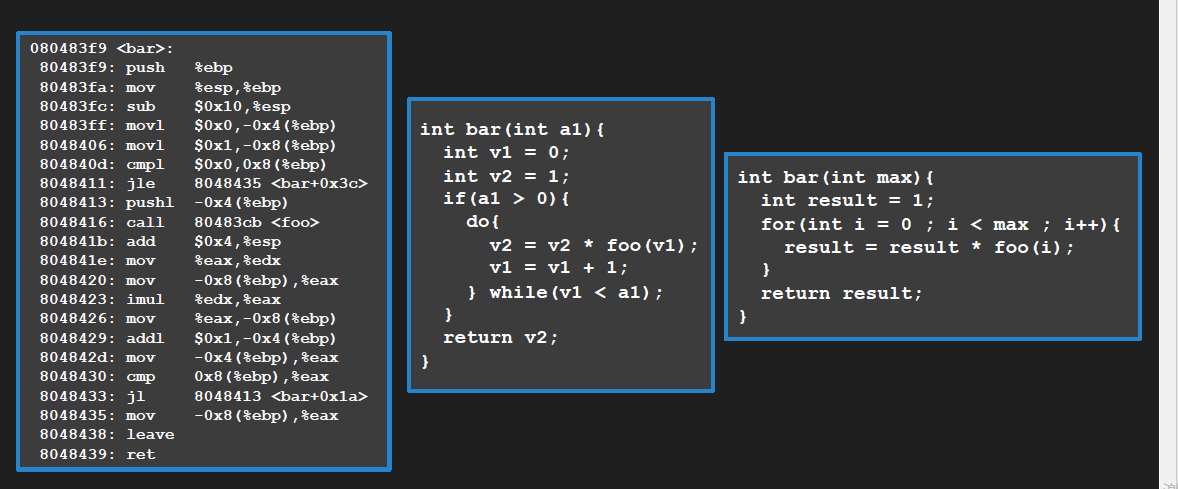
**b）复杂环路结构 Complex Loop Structure**

**3）缺乏高级语义学**

高级语义（如变量名称）在编译过程中丢失，无法由反编译器恢复，因此，反编译器通常会将默认名称分配给变量。

作为说明这些问题的示例，我们考虑如图1所示的代码，

which code would you rather analyze?

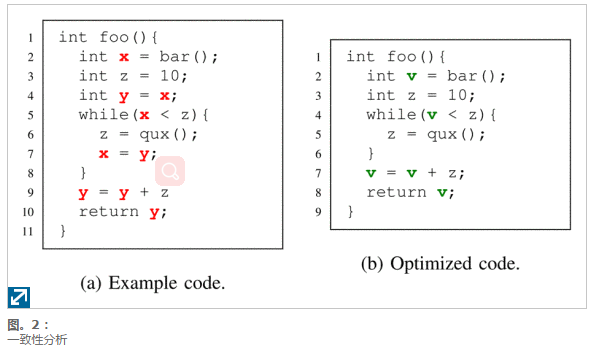


2，表达简化Expression Simplification

**A.一致性分析Congruence Analysis**

一致性分析是我们从反编译代码中删除冗余变量的方法。关键概念是识别代表相同值的变量，并且可以在保留语义的同时用单个代表变量代替。

在这个例子中，第4行和第7行在变量x和y之间复制一个值。另外，用单个代表（如变量v）替换x和y并不会改变程序的语义。此外，这种替换会导致两个可以安全删除的v = v（行4和7）的简单的复制语句，从而产生更紧凑和可读的代码，如图2b所示。不改变程序的语义



这个简单的例子说明了代码在变量一致性表征中的不同属性。总之，需要涵盖以下几个方面。

**1）相同类型Same Types**

**2）非干扰定义Non-Interfering Definitions**

**3）一致显示语句Congruence-Revealing Statements**

**B. 条件简化Condition Simplification**

此步骤的目标是在反编译代码中找到最简单的高级逻辑表达式.

简化逻辑表达式有两个方面的帮助：第一，它有助于将语义上等效的高级条件恢复到编译器发出的低级检查。第二，它有助于清除由原始代码中的错误引起的任何误解

**低级检查Low-Level Checks**

在编译期间，编译器使用一个名为*tiling*的转换将高级程序语句减少到汇编语句中。因此，每个高级语句都可以转换成一系列语义上等效的汇编指令在此过程中，高级谓词被转换为可以有效执行的语义上等效的低级检查。作为一个例子，我们考虑中所示的代码图1a。

**守则中的错误Errors in the Code**

为了提供通用的简化方法，我们将我们的技术基于Z3定理证明器。我们的做法如下。首先，我们将Dream IR中的逻辑表达式转化为Z3定理证明器的语义等效符号表达式。为了实现忠实的表示，我们将变量建模为固定大小的位向量，具体取决于它们的类型。位向量理论允许对无符号和有符号双补数运算的精确语义进行建模。在这个转换过程中，我们保持每个符号变量和原始逻辑表达式中它所代表的相应变量之间的映射。其次，我们使用定理证明来简化和规范化符号表达式。最后，我们使用映射来构建Dream IR中的逻辑表达式的简化版本。

**C.指针转换Pointer Transformation**

**指针到阵列转换Pointer-to-Array Transformation**

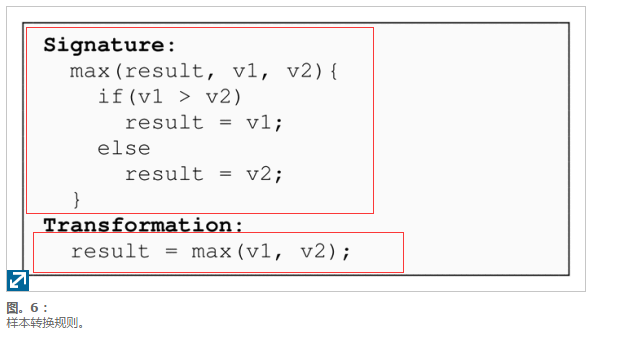
**参考指针转换Reference-to-Pointer Transformation**

3.代码查询和转换

**A.Dream IR基于逻辑的表示Logic-Based Representation of Dream IR**

**B.转型规则**

逻辑规则的一个非常强大的方面是相应的查询可以用于多个目的。例如图6显示了一个示例转换规则，用于搜索计算两个值中最大值的if语句，并通过调用max库函数来替换它们



**C.应用转型**

代码查询和转换引擎是我们后续的代码优化的基础，它们标识某些代码模式并应用相应的转换，旨在简化代码并提高可读性

4.控制流简化

we present our techniques to simplify the control flow of decompiled code简化反编译代码的控制流程

**A.循环变换Loop Transformation**

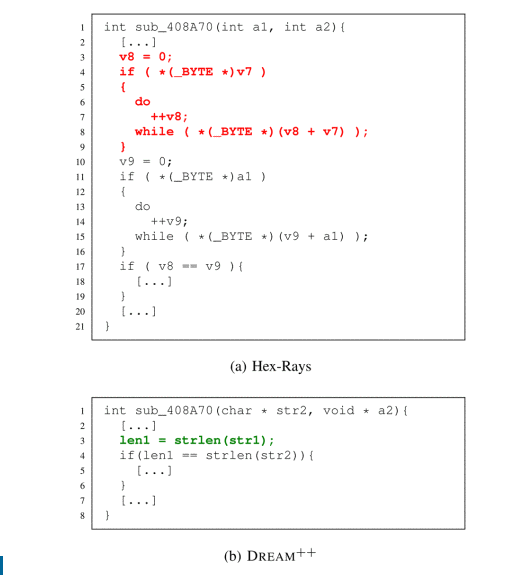
**守卫的Do-While循环**

**无保护的Do-While循环**

**功能概述**

功能内联是众所周知的编译器优化，其中所有对某些功能的调用都被替换为功能代码的就地副本。这改善了运行时性能，因为完全消除了调用和从函数返回的开销。

反向功能内联对于恶意软件的手动分析很有价值。作为一个简单的例子说明了功能概述的好处，我们考虑了图7所示的Cridex恶意软件系列的摘录代码。图7a所示的十六进制反码代码中的两个循环中的每个循环通过将每个字符的计数器递增1来计算字符串的长度，直到找到终止空字符。Dream ++将这两个块标识为strlen库函数的实现，并用相应的函数调用替换它们，如图7b所示。这个简单的例子可以深入了解功能概述代码分析的好处。



5.语义感知命名

在本节中，我们介绍了反编译代码中变量级别的可读性改进

**有意义的名字**

变量名称在分析源代码时起着重要的作用。这些名称揭示了有关变量目的的有用信息，以及如何在程序中使用变量。我们根据它们发生的上下文给变量有意义的名称。我们根据它们发生的上下文给变量有意义的名称。这里我们区分以下情况：

**标准库呼叫**

**基于上下文的命名**

**B.命名常数** 常数是逆向工程过程中的重要角色

**库API常量**C标准库和Windows API中的许多函数定义了特殊的命名常量

**文件魔数**

6.用户研究设计

**A.任务选择**

**1）加密**

编码功能广泛应用于恶意软件以及良性应用。恶意软件可以使用C＆C服务器加密交换的消息，并编码内部字符串以避免静态分析。此任务是从Stuxnet恶意软件中解析包含Stuxnet主DLL的.stub部分的函数。

**2）自定义编码**

此任务是来自Stuxnet恶意软件系列的XOR加密/解密功能。该函数使用0xAE12执行字义XOR，并被许多Stuxnet组件用于伪装一些字符串。

**3）动态解决API**

为避免静态分析，恶意软件通常避免在导入表中列出所需的API功能。相反，它可以在运行时动态解决它们。该任务是Cridex恶意软件的一个功能，它将API函数的名称作为输入，并返回相应的起始地址。

**4）字符串解析**

恶意软件经常从C＆C服务器接收命令和配置文件。因此，需要解析这些命令以从C＆C消息中提取参数和其他信息。此任务是从URLZone恶意软件注入解析函数。该函数检查序列％[A-Z0-9]％的第一次出现的字符串，并返回一个指向该字符串开头的指针及其长度。

**5）下载并执行**

一个非常常见的功能是从C＆C服务器下载可执行文件，然后执行它。例如，每次安装服务的情况就是这样的[11]。该任务涉及分析仙女座恶意软件的更新机制。该片段从远程服务器下载文件，并检查它是否是有效的PE可执行文件或包含可执行文件的Zip存档。在这种情况下，文件保存在磁盘上并执行。

**6）域生成算法**

恶意软件通常配备有域生成算法（DGA）动态生成用于C＆C域名（例如，取决于种子值，例如*作为*当前的日期/时间和Twitter趋势）[2] 。这是一种强大的技术，可以使僵尸网络对攻击和删除尝试更有弹性。该任务包含Simda恶意软件的DGA。

**B.预研**

由于恶意软件分析是一项非常复杂的任务，需要专门的技能，因此我们招募了成功完成我们的恶意软件新兵训练营的学生。恶意软件启动训练营是我们大学每个学期举办的实验课程，学生将被引入到恶意软件和二进制代码分析领域。在每个任务之后，要求参与者提供有关他们分析的代码质量的反馈

**1）预研结果**

**C.方法**

主要的用户研究几乎与预测不变。方法在以下几点上有所不同。反编译器的名称是盲目的，以免偏见参与者。另外，这项研究没有在实验室进行，而是通过我们的在线学习平台进行。这个决定是出于以下几个原因：首先，并不是所有完成恶意软件新兵训练的学生都在本地生活，我们希望最大限度地发挥我们的招聘池，因为这个级别的参与者很少。第二，我们也想与专业恶意软件分析师进行研究，期望他们来到实验室是不切实际的。我们决定在线进行整个研究，以保持结果的可比性。

**1）变量和条件**

**2）条件分配**

**3）用户感知**

**4）统计检验**

对二进制代码恢复高层次抽象的反编译方法和开发原理方法进行了大量研究。在高水平上，本文提出的工作有四条相关的研究。首先，从可执行文件中提取二进制代码的方法。第二，关于恢复源代码重建所需的抽象的研究。第三，端对端反编译器的工作。最后，查询代码库和应用转换的技术。