doi:10.3969/j.issn.1000-436x.2013.08.018

无文件系统嵌入式固件后门检测

忽朝俭[1,2]，薛一波[1]，赵粮[2]，李舟军[3]
(1. 清华大学 信息技术研究院，北京 100084；2. 绿盟科技，北京 100089；3. 北京航空航天大学 计算机学院，北京 100191)

摘 要：在无文件系统嵌入式固件中，系统代码和应用代码集成在单个文件中，无法看到熟悉的系统调用名字，故针对此类固件的分析将更为困难。以此类固件为研究对象，分析了其中的库函数识别问题，并提出了一种针对网络套接字和字符串/内存操作函数的基于启发式规则的识别方法。在此基础上，讨论了多种典型的后门类型检测问题，包括未授权侦听者、非预期功能、隐藏功能和向外的连接请求等，并在一款实际系统上成功检测出多个后门（其中有一个严重级别的）。实验结果表明，提出的针对无文件系统嵌入式固件的库函数识别方法对于此类固件的安全分析具有重要的参考价值。

关键词：嵌入式系统；固件；文件系统；库函数识别；后门检测
中图分类号：TP311.1 **文献标识码**：B **文章编号**：1000-436X(2013)08-0140-06

Backdoor detection in embedded system firmware without file system

HU Chao-jian[1,2], XUE Yi-bo[1], ZHAO Liang[2], LI Zhou-jun[3]
(1. Research Institute of Information Technology, Tsinghua University, Beijing 100084, China；2. NSFOCUS, Beijing 100089, China;
3. School of Computer Science & Engineering, Beihang University, Beijing 100191, China)

Abstract: Any embedded system firmware without file system will integrate its system code and user application code into a single file. This setting has brought some additional difficulties to analyze them. Aimed at this kind of firmware, the problem of library function identification was analyzed, and several heuristic methods to recognize some important function relevant with manipulating network socket and character string / memory were proposed. Based on this analysis, the backdoor detection problem of some typical types including unauthorized listener, unintended function, hidden function, outward connection request etc. were discussed, and several backdoors (one is critical level) in a real world firmware were found. The result shows this method of identifying library function can be useful for security analysis to this type of firmware.

Key words: embedded system; firmware; file system; library function identification; backdoor detection

1 引言

嵌入式系统是一种完全嵌入受控设备内部，为特定应用而设计的专用计算机系统。典型的嵌入式系统一般由 4 部分组成：处理器、存储器、输入输出（I/O）和软件。由于历史的原因，嵌入式系统的软件又称为固件。获取嵌入式固件的方法很多，可以直接从提供商的更新发布中获取，也可以从嵌入式设备导出（dump）。其中，前一种方法比较简单，而后一种需要较高的技术和实践能力，并可能需要借助于特定的工具。

根据嵌入式固件是否使用文件系统，本文将其分为带文件系统的（例如 VxWorks、Windows CE 和 uClinux 等）和无文件系统的（例如 ThreadX 和 uCOS-II 等）2 类。对于规模和复杂度较大的嵌入式固件，通常需要使用不同的文件来实现不同的功能模块，并使用某种特定的文件系统（例如 romfs、ext2 和 fat 等）负责存取和管理其内部的文件信息。

收稿日期：2013-05-01；**修回日期**：2013-07-18
基金项目：国家自然科学基金资助项目（61170189, 60973105, 90718017）；国家科技重大专项基金资助项目（2012ZX03002002-003）
Foundation Items: The National Natural Science Foundation of China (61170189, 60973105, 90718017); The Technology Major Project of the Ministry of Science and Technology of China (2012ZX03002002-003)

由于带文件系统的嵌入式固件通常可以根据其文件系统将其分解为不同的软件模块，然后使用针对一般计算机程序的常用分析方法对单个模块进行分析，本文不再赘述这种情况。

目前，针对无文件系统嵌入式固件的安全性分析还较零散，也不够深入。无文件系统的固件规模上通常不及带文件系统的固件庞大，容易让人误认为出现安全问题的概率不大，故其安全性分析目前尚未得到足够的重视。另一方面，无文件系统的嵌入式固件通常没有系统软件和应用软件的区分（例如，ThreadX 操作系统是一个函数库，应用程序可使用该库提供的“创建线程”函数创建一定数目的线程，每个线程完成一个任务，应用程序经编译后和该库链接，生成单一的二进制固件文件），两者集成在单个文件中，故无法看到熟悉的系统调用（库函数）名，从而使得实际的分析难度更高，目前尚未看到对其安全性的深入分析。

本文关注无文件系统的嵌入式固件（如无特别说明，下文提到的固件均指此类固件）的安全问题，重点对其中的后门检测问题进行深入分析，并对面临的主要挑战和可能的应对措施进行讨论。此类固件由于规模较小，因此出现漏洞的概率确实要小一些，但并不代表其不会被故意植入后门。一般的计算机程序运行于通用的硬件平台之上，因此不管是使用物理环境还是采用虚拟化技术模拟执行环境都相对比较容易。而固件通常专用于特定的硬件设备，因此不管是采用物理环境还是模拟环境，最终将固件运行起来都要困难很多。更严重的是，针对一般的计算机程序运行过程的监控（用于获取程序的输出或者内部状态等信息）很容易实现，而监控不同提供商的嵌入式设备通常需要使用其配套的调试设备，因此实现通用的监控机制难度较大。基于上述观察，本文对无文件系统固件后门检测问题采用静态分析为主、动态分析为辅的方法。

2　相关工作

针对后门检测，通常采用动态监控或静态扫描的方法。例如，ZHANG等[1]提出并实现了一种通过被动地监视站点的Internet访问链接，并基于分组的大小和时间特性检测交互流量的通用算法，可检测运行在非标准端口的标准服务和在标准端口运行的非标准服务后门。HORNG等[2]提出了一种基于动态链接库注入技术来记录所有动态链接库的网络交互情况，从而识别后门的方法。而静态扫描的方法主要用于反病毒软件，通过预先提取后门签名，然后采用模式匹配技术确认程序中是否隐藏有后门代码。WYSOPAL和ENG[3]讨论了几种典型的后门类型，并介绍了几种基于启发式规则的静态检测方法，本文对固件后门的检测方法主要受该工作的启发。除了单纯的动态或静态，也有混合使用动静态方法的。DAI等[4]提出了一种消除响应可计算认证系统中后门的框架，通过分解认证模块为简单逻辑组件和加密/模糊变换组件，并分别采用代码正确性分析和沙箱测试的方法进行验证。另外，ELHAM等[5]还提出了一种使用人工神经网络和遗传算法预测后门存在可能性的方法，而HOMMES等[6]将激活隐秘后门的端口敲击序列简化为关联规则挖掘问题。

不管是通过人工逆向还是自动代码分析来剖析和理解二进制代码的行为，有意义的系统调用名字通常可大大降低分析的难度。但在无文件系统嵌入式固件中，无法看到熟悉的系统调用名字，故如何识别重要的库函数将是此类固件后门检测亟待解决的问题。

对于Windows PE文件，通过解析导入地址表（IAT, import address table）、IDA Pro和Ollydbg等工具均实现了对系统调用函数的识别（动态链接的运行时库中函数与之类似，不再赘述），可使用有意义的字符串表示每个系统调用函数的名字。除此之外，IDA Pro中还实现了一种称为快速库辨认与识别的技术(FLIRT, fast library identification and recognition technology)[7]来识别对静态链接运行时库中的函数调用。FLIRT的基本思想是对主要编译器（例如Microsoft或Borland的各种编译器以及GNU的GCC等）静态链接运行时库中的函数，取其指令序列的前 32 byte作函数签名（带有简单的冲突处理机制），并保存为不同的签名库文件。当IDA Pro反汇编二进制代码时，通过签名匹配技术就可识别已知的库函数，简化反汇编工作，进而简化控制流图和调用图的构造。值得注意的是，IDA Pro的FLIRT签名的制作过程并不是一个完全自动化的过程，当多个函数在预定的签名生成算法下产生签名碰撞的情况下，通常需要手工干预。

通常需要利用反汇编技术[8,9]将固件的二进制代码转换为人工容易理解的汇编指令。二进制反汇编主要包括线性扫描（linear sweep）和递归遍历（recursive traversal）2 种。Ollydbg和Objdump工具

是使用线性扫描反汇编方法的典型代表。Hex- Rays公司的IDA Pro是一款基于递归遍历算法且准确率相当高的商业反汇编工具。但由于这 2 种方法均存在一定的局限性，故对于典型的二进制代码，到目前为止，还没有一款反汇编工具能达到 100%的准确率和覆盖率。

3 库函数识别

根据上文的分析，库函数识别对于后门检测的成败具有重要的决定作用。实际上，获取准确、完整的反汇编代码不仅对库函数识别具有重要意义，而且在后门检测中同样扮演重要角色。线性扫描反汇编可能取得极高的覆盖率，但其会对任何嵌入在代码中的数据也进行反汇编，而识别所有嵌入在代码中的数据又相当困难，故其准确率通常相当低。基于上述考虑，通过对准确率较高的递归遍历反汇编工具进行优化，提高其覆盖率将是一种可行的方法。

3.1 反汇编的覆盖率问题

本文以递归遍历反汇编工具 IDA Pro 为基础，考虑如何使用静态方法提高反汇编的覆盖率。可能的应对措施包括：基于数据流分析的反汇编优化、基于序言/结束语的函数识别方法。

1) 基于数据流分析的反汇编优化。递归遍历的主要局限性在于遇到间接跳转/调用时，难以确定最终的目的地址/函数，从而造成大量代码不能识别并反汇编。通过使用常量传播等数据流分析算法可以确定部分间接跳转、调用的目标地址/函数，从而提高反汇编的覆盖率。

2) 基于序言/结束语的函数识别。传统的递归遍历通常难以识别被间接调用的函数。对于源码中的函数，主流编译器生成的二进制代码通常会具有一些显著的特点，例如函数序言和函数结束语。函数序言是出现在函数代码开始处的指令，函数结束语是出现在函数结束处（返回到调用者之前）的指令。通过搜索特定的函数序言和与之匹配的函数结束语，是发现被间接调用函数的一种简单有效措施。

3.2 库函数识别问题

本质上，无文件系统的嵌入式固件中也存在库函数。其系统调用函数通常以静态链接的库函数形式出现，两者指代的对象相同（如无特殊说明，本文提到的库函数即系统调用函数）。但由于这种类型的固件通常就是单个文件，因此通常无法简单地区分库函数与普通函数。这种情况不仅对本文的后门识别研究带来直接的挑战，而且对更广义的漏洞检测也有巨大的影响。

基于上述观察，本文提出了一种针对网络套接字和字符串/内存操作函数的基于启发式规则的检测方法，其基本工作流程如图 1 所示。

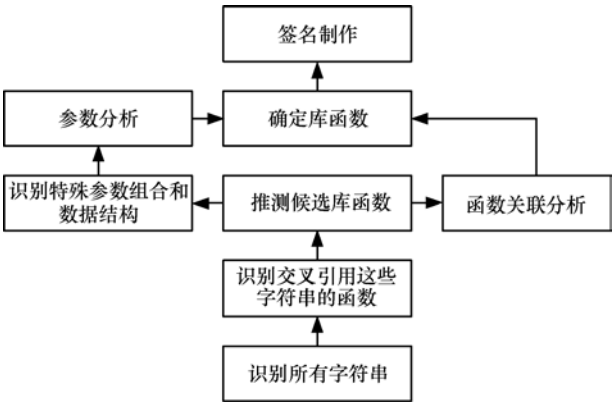


图 1 库函数识别的工作流程

其中，本文使用的主要库函数识别启发式规则包括：基于字符串交叉引用的候选库函数推测和基于参数和函数关联分析的库函数确认。

1) 基于字符串交叉引用的候选库函数推测

程序员使用字符串操作函数时，有可能会使用到常量字符串。而为了增加代码的可读性或者单纯的调试目的，调用库函数后也可能对返回值（或者输出参数）进行判断，并且在失败的情况下使用打印函数输出特定的信息，这些信息通常也是硬编码的字符串。通过对感兴趣字符串的交叉引用进行分析，不难初步推测出部分候选库函数。例如：某固件中调用 recv 函数之后，判断失败的情况下会打印“recv fail:…”形式的字符串；而通过对交叉引用“QUIT”的几个函数的分析很容易确认 strcmp。

2) 基于参数和函数关联分析的库函数确认

某些重要函数可能带有特定数目的参数，且通常的情况下这些参数的值或者来源是固定的。基于该观察，可以很容易识别出这些库函数。例如：调用socket函数时，常用的参数为(2, 1, 0)和(2, 2, 0)，分别表示创建TCP和UDP套接字；创建socket之后，通常随后会以socket函数的返回值作为参数调用bind或者connect函数，这时候通常还会填充一个sockaddr结构，该结构中的第 2 个成员表示端口号（例如：Modbus协议[10]的 502 端口号是作者感兴趣的）。

4　后门检测

在计算机科学中，后门特指对计算机系统的未授权访问，是一种严重的安全隐患。需要注意的是，后门既可能是硬件上的，也可能是软件上的。软件形式的后门是指隐藏在程序中（由程序员所创建）的、可在用户未知或者未同意的情况下提供对系统非法访问的代码。本文重点关注未授权侦听者、非预期功能、隐藏功能和向外的连接请求 4 种常见的固件后门（软件形式）的检测问题。未授权的侦听者是指规范中未规定要使用的端口，非预期的功能是指具体实现与规范描述不符的功能，隐藏功能是指规范中未指定的功能，而向外的连接请求是指主动连接外部节点的网络操作。

由于嵌入式系统对实时性的要求比较高，针对固件后门的检测和防护不应过多依赖于运行时的监控信息，因此本文采用动静结合的方式，如图 2 所示。

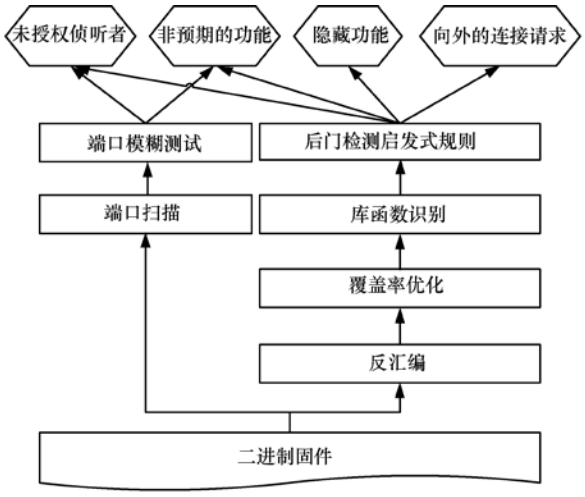


图 2　动静结合的固件后门检测整体工作流程

其中，静态方式主要通过对 IDA Pro 生成的反汇编代码执行“覆盖率优化”和“库函数识别”的基础上，采用“启发式规则”+“逆向分析”的方法；而动态分析主要依赖“端口扫描”和“端口模糊测试”技术。

4.1　后门检测启发式规则

针对上述不同的后门类型，本文分别提出了有针对性的后门检测启发式规则，包括以下 4 部分。

1) 未授权的侦听者

对 bind 函数作交叉引用分析和调用参数分析，确定固件中使用的所有未授权端口；并对 recv、recvfrom、send 和 sendto 函数作交叉引用分析，确定这些未授权端口的网络交互行为。

2) 非预期的功能

对 recv 和 recvfrom 函数作交叉引用分析，确定授权端口的消息处理循环。针对规范指定的每种功能，在消息处理循环中确定其处理逻辑，并对处理逻辑进行逆向分析，找出实际功能与其规范的差异。

3) 隐藏功能

对 recv 和 recvfrom 函数作交叉引用分析，确定各端口各自的消息处理循环。针对出现在消息处理循环中的多余处理逻辑（而规范中并未描述）进行逆向分析，理解其具体功能。

4) 向外的连接请求

对 connect、gethostbyname 和 gethostbyaddr 函数作交叉引用分析和调用参数分析，确定固件中所有向外的连接请求使用的目的地址和端口；并对 recv、recvfrom、send 和 sendto 函数作交叉引用分析，确定这些未授权端口的网络交互行为。

4.2　端口模糊测试

端口模糊测试是指对特定端口执行的模糊测试。一般可以先采用端口扫描工具（例如Nmap）确定系统打开的端口（侦听端口）。若存在打开的未授权端口，则检测到未授权的侦听者。再针对打开的授权端口（例如Modbus的TCP 502）进行模糊测试[11,12]。端口模糊测试可检测非预期功能类型的后门，其具体检测流程如下。

1) 根据使用不同端口的协议（例如 Modbus 的 TCP 502）自身的特点，选用特定的模糊测试数据生成器和异常检测规则。

2) 将生成器生成的数据发送到协议授权使用的端口，并记录每个请求和相应的响应数据。

3) 根据异常检测规则对请求/响应交互序列进行过滤，找出实际响应与协议规范描述存在偏差的请求以及未收到任何响应的请求。

4) 人工对过滤后的请求/响应交互序列进行细致分析。

对于打开的未授权端口，也可以执行模糊测试，并记录每个请求和相应的响应数据做进一步分析。但由于缺乏对端口的先验知识，效果通常不容乐观。

5　实例分析

本节以一个完整的例子验证和演示上述后

门检测方法的实际应用过程和检测效果。其中，嵌入式设备选用霍尼韦尔的 HC 900 远程终端系统，固件为原始文件（无文件头，指令从首字节开始），基于 ARM 架构指令集，采用大头序，版本为 4.4。

5.1 反汇编覆盖率

目前有多种反汇编工具可以对其进行反汇编。本文采用 IDA Pro 作为基础反汇编工具。通过观察发现，HC 900 固件中的绝大多数函数均有规则的序言和结束语（“STMFD”和“LDMFD”）。基于该观察，使用基于序言/结束语的启发式函数识别方法，对其覆盖率进行的二次优化实验，结果如表 1 所示。基于数据流分析的反汇编优化方法目前仍在编码阶段，暂无实验数据。

表 1 提高反汇编覆盖率的启发式规则

启发式规则	#识别的函数	覆盖率/%
IDA Pro	2 804	75.1
序言/结束语	3 627	97.2
优化效果	823	29.4

其中，覆盖率=#识别的函数/ #确认的总函数。总函数数量为 3 731，其是通过人工逆向分析结果估计。具体实施方法为：1）由人工确认未反汇编的字节片段是否是数据；2）然后在所有非数据的字节片段执行递归遍历反汇编；3）通过几次迭代，当无法继续时，停止反汇编，并假定最终识别的函数数量即为程序中总的函数数量。该方法仍可能将某些数据识别为代码，但由于是人工参与，故引入错误的概率极小，将其作为估算固件中总函数数量的依据是合理的。

5.2 库函数识别

熟悉的系统调用和库函数名字对于增加程序的可读性、降低分析难度具有积极的作用。本文重点关注网络套接字、字符串/内存操纵函数的识别，采用的方法主要有字符串交叉引用、参数分析和函数关联分析 3 种。识别的部分主要函数和识别的方法如表 2 所示。

识别上述函数的过程中，作者发现了明显的静态链接库函数的特征：网络套接字函数位于一段连续的代码区域，而字符串和内存操作函数位于另一段代码区域（来自 2 个不同的静态链接库）。基于该发现，作者对这些函数制作了 FLIRT 签名。在对 HC 900 之前版本（4.3、4.2、4.1）固件分析的过程中，这些函数签名均得到了匹配，再次验证了其库函数的特性。

5.3 后门检测

针对 HC 900 的 4.4 版本固件，根据上述检测方法，对未授权侦听者、非预期功能、隐藏功能和向外的连接请求 4 种典型的后门类型进行了检测和分析。实验结果显示：该固件中未发现向外的连接请求，但存在未授权侦听者、非预期功能、隐藏功能 3 种类型的后门，具体情况如下。

1) 使用 Nmap 扫描的结果显示：除 Modbus 协议规定使用的 TCP 502 端口外，HC 900 还打开了 TCP 和 UDP 的 7777 端口。利用静态启发式规则的分析同样验证了该结果。

2) 采用“端口模糊测试”对 TCP 502 端口进行测试的结果显示：0x14 和 0x15 功能与协议规范描述存在较大偏差。通过对固件 TCP 502 端口命令处理循环的逆向分析显示，该固件仅实现 Modbus 协议规定的部分功能码：0x01−0x06、0x08、0x10、0x11、0x14 和 0x15。

对该命令处理循环中每个功能码对应的处理逻辑的进一步分析显示：0x14 和 0x15 2 个功能码并没有实现规定的功能；尤其是 0x15 的实现存在严重后门，通过发送精心构造的数据，可以造成 HC 900 系统停止采集数据，并向用户发送攻击者设定的数据，如图 3 所示。

3) 对 TCP/UDP 7777 端口的命令处理循环的逆向分析显示：TCP 7777 端口接受“R”（写入）、“T”（读取）和“X”，UDP 7777 端口接受“B”和“T”命令。

表 2 识别的库函数

函数	类型	字符串交叉引用	参数分析	函数关联分析
accept, bind, connect, getsockname, getsockopt, listen, select, setsockopt, socket	网络套接字		√	√
recv, recvfrom, send, sendto	网络套接字	√		
malloc, memset, memcmp, memmove, memcpy	内存		√	√
strcat, strchr, strcmp, strcpy, strcspn, strlen, strncat, strncmp, strncpy, strspn, strstr	字符串	√		

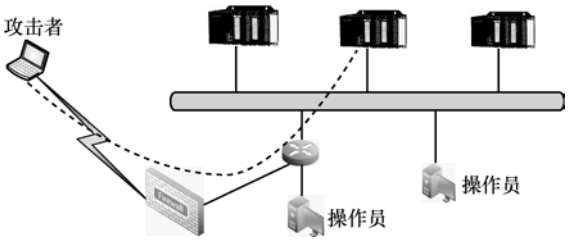


图 3　HC 900 系统 0x15 实现后门

6　结束语

嵌入式固件与计算机软件运行环境的差异，使针对固件的分析通常采用静态的方法。而无文件系统嵌入式固件与带文件系统的嵌入式固件在库函数上的显著差异又使得对其进行静态分析的难度更大。

本文以二进制无文件系统嵌入式固件为研究对象，讨论了其中的库函数识别和后门检测问题，成功地识别了一款真实固件中主要的网络套接字、字符串/内存操纵函数，并检测到多个后门（包括一个严重级别的后门）。实验表明，本文提出的针对无文件系统嵌入式固件的库函数识别方法对于该类固件的安全性分析具有重要的参考价值。

参考文献：

[1] ZHANG Y, PAXSON V. Detecting backdoors[A]. USENIX Security Symposium[C]. Denver, Canada, 2000. 12.

[2] HORNG S J, SU M Y, TSAI J G. A dynamic backdoor detection system based on dynamic link libraries[J]. International Journal of Business and Systems Research, 2008, 2(3):244-257.

[3] WYSOPAL C, ENG C, SHIELDOS T. Static detection of application backdoors[J]. Datenschutz und Datenschutz-DuD, 2010, 34(3):149-155.

[4] DAI S F, WEI T, ZHANG C, *et al*. A framework to eliminate backdoors from response computable authentication[A]. IEEE Symposium on Security and Privacy[C]. San Francisco, USA, 2012. 3-17.

[5] ELHAM S, NARGES A. Backdoor detection system using artificial neural network and genetic algorithm[A]. International Conference on Computational and Information Sciences[C]. Chengdu, China, 2011. 817-820.

[6] HOMMES S, STATE R, ENGEL T. Detecting stealthy backdoors with association rule mining[A]. International IFIP TC 6 Conference on Networking[C]. 2012. 161-171.

[7] GUILFANOV I. FLIRT[EB/OL]. http://www.hex-rays.com/idapro/flirt.htm.

[8] SCHWARZ B, DEBRAY S, ANDREWS G. Disassembly of executable code revisited[A]. IEEE Working Conference on Reverse Engineering[C]. Richmond, USA, 2002. 45-54.

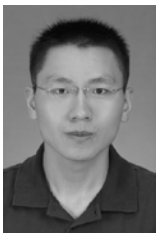
[9] KRUEGEL C, ROBERTSON W, VALEUR F, *et al*. Static disassembly of obfuscated binaries[A]. USENIX Security Symposium[C]. San Diego, USA, 2004. 255-270.

[10] Modbus IDA[EB/OL]. http://www.modbus.org/specs.php.

[11] TAKANEN A, DEMOTT J, MILLER C. Fuzzing for Software Security Testing and Quality Assurance[M]. Artech House Inc, 2008. 22-32.

[12] DURAN J, NTAFOS S. An evaluation of random testing[J]. IEEE Transactions on Software Engineering, 1984, 10(4):438-444.

作者简介：



忽朝俭（1982-），男，河南南阳人，清华大学&绿盟科技在站博士后，主要研究方向为程序分析、软件漏洞检测和网络安全。

薛一波（1967-），男，山东莱阳人，博士，清华大学研究员、博士生导师，主要研究方向为计算机体系结构、并行处理和网络安全。

赵粮（1969-），男，山东宁津人，博士，绿盟科技首席战略官，主要研究方向为网络安全和云安全。

李舟军（1963-），男，湖南湘乡人，博士，北京航空航天大学教授、博士生导师，主要研究方向为网络与信息安全、数据挖掘与文本挖掘。