Operating systems and software in general, continuously grow in size and complexity. As a result, software contains programming errors that frequently allow attackers to gain illegitimate access, and even fully control systems. In the past, we witnessed large scale infections from worms such as CodeRed, Blaster, and Sasser that managed to infect hundreds of thousands of hosts, while the Slammer worm exhibited phenomenal speed in infecting almost every vulnerable server in minutes. More recently, we saw attackers exploiting bugs in popular applications, such as web browsers, to take control and organise compromised systems into large collections of networks that are used for sending SPAM, carrying out distributed denial of service (DDoS) attacks, and extracting personal information (credit card numbers, passwords, etc).

Current solutions have been able to alleviate the problem partially, but in practice have proven inadequate in detecting attacks and generating countermeasures in a timely manner. This dissertation addresses the problem of automatically and reliably detecting previously unknown attacks, and generating vaccines that can deter new infections in their early stages. We present three novel ways of using virtualization to detect zero-day attacks, and automatically generate countermeasures. Our solutions are based on a technique called dynamic taint-analysis, used to capture the most prominent self-propagating attacks. Most importantly, they apply to legacy hardware and software, and generate no false positives.

Dynamic taint analysis is based on tracking possibly dangerous data such as network data (explained in detail in Section 2). Network data are practically never used to directly control the flow of a program. For instance, network values are not used as function pointers. Attackers frequently exploit memory corruption errors to overwrite program values controlling execution flow such as function pointers. Dynamic taint analysis detects when network data are used in this way, thus identifying exploit attempts. Implementing this technique in software requires a virtualization layer, such as an emulator or a dynamic binary translation framework, and usually incurs a massive performance overhead in the range of 1000%-2000%. This thesis will attempt to take on the challenges involved with applying dynamic taint analysis on existing systems. Our goals can be summarised in the following research questions:

- Can we find solutions for detecting zero-day attacks, by means of dynamic data-flow tracking, in unmodified software, and without requiring access to source code or specialised hardware?
- Can we mitigate the performance overhead imposed by dynamic data-
flow tracking to scale our solutions to varying computing systems, such as servers, desktops, and smartphones?

The contributions of this thesis are presented in Chapters 3 to 5:

**Argos** Chapter 3 presents a secure emulator named Argos. Argos is a platform for the next generation high-interaction honeypots that automates the procedure of capturing zero-day attacks, and generates a simple “vaccine” for deployment on NIDS. It offers whole-system protection in software by way of a modified x86 emulator which runs our own version of dynamic taint analysis that is able to protect any (unmodified) OS including all its processes, device drivers, etc. Argos takes into account complex memory operations, such as memory mapping and DMA that is commonly ignored by similar projects. It is able to detect attacks such as buffer overflow and format string / code injection exploits, and trigger alerts that result in the automatic generation of signatures based on the correlation of the exploit’s memory footprint and its network trace. After an attack is detected, we inject OS-specific forensics shellcode to extract additional information on the exploited code. Finally, by comparing signatures from multiple sites, we refine the generated signatures automatically and auto-distribute them to remote intrusion detection and prevention systems (IDS and IPS).

**Eudaemon** In Chapter 4 we developed a technique that transparently enables desktop systems to securely act as honeypots. Eudaemon aims to blur the borders between protected and unprotected applications, and brings together honeypot technology and end-user intrusion detection and prevention. It is able to attach to any running process, and redirect execution to a user-space emulator that will dynamically instrument the binary by means of taint analysis. While the target application is emulated, any attempts to subvert control flow, or to inject malicious code will be detected and averted. When desired Eudaemon can reattach itself to the emulated process, and return execution to the native binary. It can move an application between protected and native mode at will, e.g., when spare cycles are available, when a system policy ordains it, or when it is explicitly requested. The transition is performed transparently and in very little time, thus incurring minimal disturbance to an actively used system. Similarly to Argos it does not require access to source code or explicit OS support, and is is able to generate signatures for NIDS.
Marvin  In Chapter 5 we address the problem of protecting light-weight devices such as smartphones by delegating security checks to a loosely synchronised replica. Smartphones have come to resemble PCs in software complexity, with complexity usually leading to bugs and vulnerabilities. Moreover, they are increasingly used for financial transactions and other privacy-sensitive tasks, becoming attractive targets for attackers. At the same time, smartphones are quite different from PCs in terms of resource constraints imposed on the design of protection mechanisms, as battery power is an extremely scarce resource. As a consequence, security solutions designed for PCs are not directly applicable to smartphones, as they may reduce battery life-time significantly. By outsourcing security checks we enable the application of heavy-weight security checks such as dynamic taint-analysis, and at the same time transparently offer backup functionality. We implemented a prototype called Marvin on the HTC Dream / Android G1 phone platform, and show that the overhead, in terms of computation and power consumption, is acceptable.