The WebJob Framework:
A Generic, Extensible, and Scalable Endpoint Security Solution

Andy Bair and Klayton Monroe
KoreLogic Security

June 2015

Abstract

The WebJob framework is a next generation endpoint security solution that, from a centralized management location, can execute virtually any program on any number of client systems at any time. This framework was designed to be generic in nature, operating system agnostic, minimal in footprint, minimal in resource consumption, and highly secure.

The framework has been deployed in a number of production environments including the Federal government and Fortune 500 businesses to perform a myriad of functions on a wide array of end systems. There is virtually no limit to the functions the framework can execute on end systems. For example, the framework has been used to perform evidence collection, enterprise searching, incident response, live forensics, system management and monitoring, and grid computing. The framework is well-suited to be the next generation endpoint system security solution.

The Problem

There are many agent-based solutions in the market today that provide important value to organizations. While the individual solutions are useful, as a whole they can complicate security and interoperability.

- Many solutions are narrowly focused on a single problem domain (so called “point solutions”). By design, this limits their effectiveness in other domains and often makes it difficult to perform custom or ad hoc tasks.
- Point solutions result in the need for multiple agents on each endpoint. This can cause operating conflicts and become problematic in older disconnected, intermittent, or low-bandwidth (DIL) environments, where adding and maintaining agents is difficult.
- Many solution frameworks lack sufficient security on the server and the endpoints. This increases the attack surface of each endpoint and exposes additional attack vectors, which could leave organizations open to prolonged and/or large-scale attacks.
- Finally, a number solutions are proprietary in nature and therefore are difficult to adapt to other problem sets or integrate into other systems or solutions.

The Solution: WebJob Framework

The WebJob framework[1] is a next generation endpoint security solution that, from a centralized management location, can execute virtually any program on any number of client systems at any time. This framework was designed to be generic in nature, operating system agnostic, minimal in footprint and resource consumption, and highly secure.

The framework centralizes data collection across all clients enabling personnel to analyze and respond to the results effectively and efficiently. The framework allows execution of regular “canned” jobs and arbitrary “ad hoc” jobs allowing organizations to adapt quickly to the

1 http://webjob.sourceforge.net/WebJob/index.shtml

changing needs of their environment or the situation at hand (e.g., emergency change requests, incident response requests, etc.). *And because the framework was designed this way, there is virtually no limit to what it can do on end systems.*

The framework is an open source client-server system that enables engineers to run arbitrary programs on end systems and aggregate the results on designated collection servers (usually the management server itself). The framework allows execution of any program on a wide array of operating systems (e.g., Linux®, Mac OS®, Windows®, Android®, etc.). It provides the automation and security to support a variety of activities including system/integrity monitoring, enterprise search, configuration management, compliance verification, automated analysis, and other activities normally requiring discrete solutions.

Figure 1 shows the key components of the framework and are described below.

- *Server* – The Server functions as the command and control or "nerve center" of the framework. All tasks are managed, scheduled, and processed from this system (or cluster of systems).
- *Client* – The Client is a lightweight program (typically less than a few megabytes) that is deployed to each participating end system and executes jobs assigned to it by the Server.
- *Job* – Jobs are programs or scripts responsible for completing a given task. A job can perform a simple action like providing a "heartbeat" indicating that the end system is alive and well, or it can perform more complicated actions such as replacing system files that have drifted from a defined standard, become corrupt/infected, or just need to be updated.

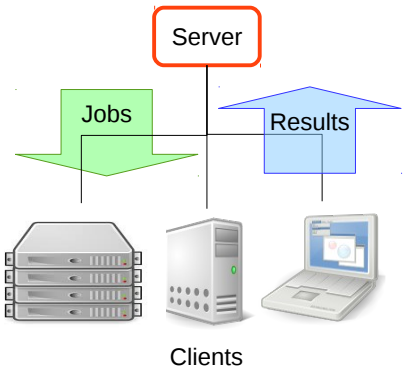

Figure 1: The WebJob Framework

The Framework in Action

The framework has been deployed in many production environments including the Federal government and Fortune 500 businesses to perform a myriad of functions on a wide array of end systems. The following are just a few examples of the framework in action.

- *"The Swiss Army Knife"* – The framework has been deployed at a Federal agency for several years in both classified and unclassified environments. Over a seven year period, the framework supported hundreds of Clients, and at one point, the framework supported 7,000 jobs per hour (or about 5 million jobs per month). The framework has performed many functions on end systems including execution of the DISA Security Readiness Review (SRR) scripts, end system compliance checking and auditing, file system searching, software management, user and group management, log management, file system integrity monitoring, process monitoring, network monitoring, disk utilization monitoring, and system health monitoring.

- *Incident Response / Live Forensics* – The framework was used in a Federally Funded Research and Development Company (FFRDC) to collect forensic information on Windows® systems suspected of having Microsoft Word documents infected with malware designed to install a back door. A custom Windows® installer program was created to harvest and upload forensic data from end systems then clean-up after execution resulting in a temporary framework of Clients. The quick response combined with data aggregation allowed security personnel to triage machines and address legitimate system infections quickly.
- *Evidence Collection / Enterprise Search* – The framework has been used in a quick response action to a data breach and extortion case for Fortune 500 companies. The framework was used to search for Personally Identifiable Information (PII) and harvest files on hundreds of production UNIX® servers and thousands of Windows® desktops/laptops where COTS tools were not equipped to conduct searches or perform collection and ad hoc tasking in a manner that met the customer's strict requirements.
- *Compliance* – The framework was used at a Fortune 500 company to search approximately 225 million files across hundreds of production UNIX® servers for sensitive unprotected data to address regulatory compliance violations. The framework allowed managers to locate the violations quickly, with pinpoint accuracy, and determine the magnitude of the problem. The rapid response (roughly three weeks) coupled with supporting data showing the issue had been resolved enabled auditors to grant compliance before the end of the calendar year.
- *System Monitoring and Management* – The framework was used to manage enterprise Intrusion Detection Systems (IDS) for a Fortune 500 critical infrastructure telecommunications firm. The framework performed a number of ad hoc tasks above and beyond its primary focus, which was IDS management. Additionally, it was used to collect, manage, and process NetFlow data.
- *Grid Computing* – The framework is currently used by KoreLogic to provide a distributed computing grid in support of activities such as malware analysis and password cracking. In the case of password cracking, for instance, the framework routinely manages multiple concurrent work orders, consisting of thousands of individual tasks, across tens to hundreds of GPU/CPU cores. This effectively makes it possible, in some cases, to condense decades of computing work into a few hours or a single day. The framework was also used to run and manage a distributed malware sandnet for the purpose of testing the performance of various Anti-Virus (AV) software products against a large set of malware samples.

Next Generation Endpoint Security

The framework can execute virtually any job on an arbitrary number of end systems with a high degree of confidentiality, integrity, and attribution. As the demand for capacity grows (either because each end system is running more jobs or because more end systems must be supported), the framework can be scaled horizontally, vertically, or both as needed.

Interoperability – Since the Client is relatively small and written in C, the framework supports a broad range of platforms (e.g., mobile devices, laptops, desktops, servers, etc.) and operating systems (e.g., Linux®, Mac OS®, Windows®, *BSD®, Solaris®, HP-UX®, AIX®, Android®,

etc.). Additionally, the framework supports thick, thin, and virtual endpoint systems as well as real/virtual servers. The framework can be configured to process job data in a number of ways ranging from basic aggregation to normalization to lightweight analysis to heavy post-processing and data correlation. By extension, this implies that the framework would be an ideal feed for larger Security Information and Event Management (SIEM)[2] systems or centralized log management systems.

Scalability – The framework is designed to scale horizontally and vertically as shown in Figure 2. A single server can support hundreds or thousands of disparate end systems depending on their polling frequencies and the total number of jobs that must be executed per unit time. Servers can also be configured as clients to other servers thereby creating a multi-tiered framework. This provides a number of unique benefits including independent or autonomous operation, alignment with chain of command, data compartmentalization, and better resource utilization. If deployed in a mesh configuration where each endpoint is multi-homed, single points of failure within the framework can be eliminated thereby improving overall survivability.

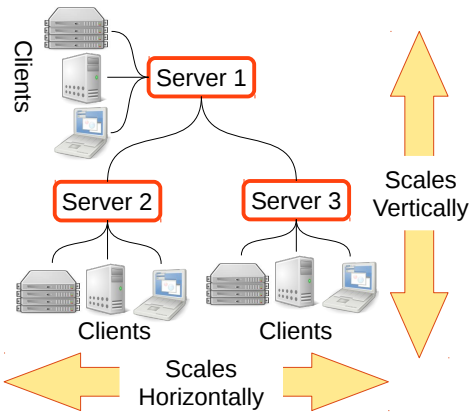

Figure 2: Framework Scalability

Utilize OS security solutions – The framework fully supports utilization of the underlying OS security solutions available. For example, the framework can be used to manage and monitor existing firewall configurations on client systems. It can also be used to enable various host-based security controls such as file permissions, groups, logging policies, access lists, etc.

Enhance OS security solutions – The framework enhances the underlying OS security solutions. For example, the framework can run anything on client systems including existing security tools to monitor critical system components such as processes, file system integrity, registry hive integrity, and general security health. The data collected from end systems can be processed on the server and alerts can be sent in response to anomalous and/or malicious activity. Alternately, the data can be minimally processed on the server and forwarded on to an existing SIEM or log analysis system for further analysis.

Reduce attack surface – The framework was designed with a security perspective and built from day one to minimize the available amount of attack surface in a number of ways.

- First, the standard Client does not have a listening daemon (i.e., it does not listen and respond to network traffic). Instead, the Client employs a pull-based model to poll the Server on a regular or ad hoc basis to receive tasks. This is significant because it eliminates an entire class of attacks. More specifically, it means that a standard Client could not be overrun by a network-based attack directed at the end system on which it runs.
- To thwart sophisticated man-in-the-middle (MITM) attacks, the standard Client supports mutual certificate authentication and digitally signed jobs. This has the added benefit of providing confidentiality, integrity, and attribution for Client/Server communications, and it ensures that the

2 http://en.wikipedia.org/wiki/Security_information_and_event_management

Client cannot be instructed to run unauthorized jobs. If configured properly, even an attacker with system privileges on the Server would have difficulty coercing a Client to execute a job that has been maliciously altered without also having access to the private job-signing key and/or system privileges on the end systems.

- To protect the confidentiality and integrity of job data, either the Server or the jobs running through the Client on the end system can be configured to encrypt job output using public/private encryption (e.g., Pretty Good Privacy) such that the Server simply becomes a holding pen for data. This technique could be used to prevent an attacker who has gained privileged access to the Server from viewing potentially sensitive data.

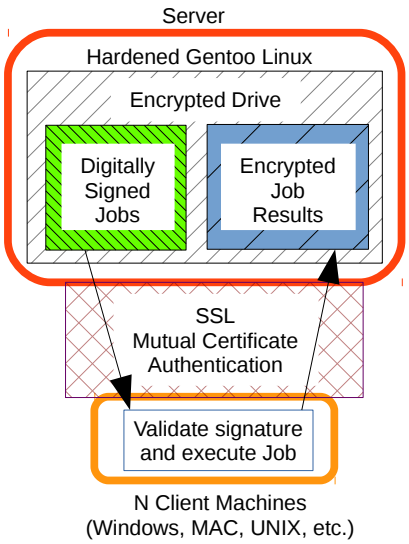

Figure 3: Framework Security

Changing connectivity – The framework supports both low- and high-agility environments, and disconnected, intermittent, or low Bandwidth/High Latency (DIL) environments. Since the Client periodically polls the Server for tasks to perform, the connection can be severed for an extended period of time with little or no ill effect. In other words the Client will simply re-establish its connection as soon as network conditions allow it, and any pending tasks can then be requested and performed. This allows client systems to temporarily "drop-off" and return at a later time depending on connectivity. Also, the framework is capable of supporting high- and low-bandwidth environments. For example, a simple heartbeat job can consume less than 1 kilobyte of bandwidth. In the case where bandwidth is limited, other utilities can be used in conjunction with the Client to stream data back to the server in a controlled fashion.

Minimize operational overhead – The framework is intentionally designed to be open-ended and flexible to reduce management and training costs. Because of this, there are any number of capabilities not yet developed that the framework could support. While it would require some development effort to engineer and integrate a new capability into the framework, this effort is typically a one-time cost. More importantly, the Return on Investment (ROI) for this effort is often realized in as few as 100 jobs run on a single end system, a single job run on 100 end systems, or some combination thereof. This implies that time and money will be saved for each and every end system serviced by the new capability beyond that break-even threshold. For environments with thousands of end systems, the savings would be significant.

Conclusion

The framework is a proven technology that has been used in numerous deployment scenarios. It has an extremely small footprint, low overhead, and supports multiple platforms and DIL environments. WebJob has been deployed in a number of US Government and Fortune 500 environments to perform a wide variety of tasks. It is both flexible and extensible, providing a single-agent solution in situations where existing COTS tools would be inadequate or impractical.