Leaf Linux

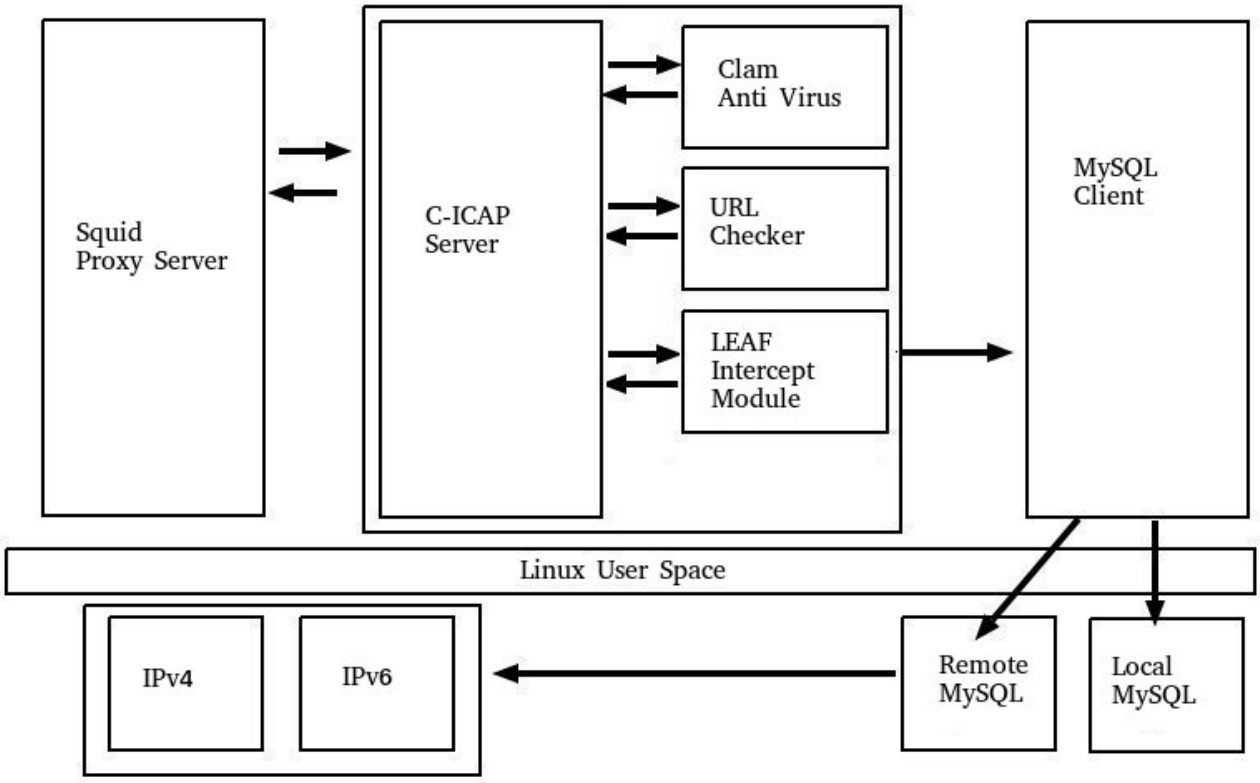
Overview

Prepared for Jan Newman
Sagecreek Partners

Document v1.3

Prepared by Jeff Merkey
455 South 940 West Orem, Utah 84058
385-299-2437

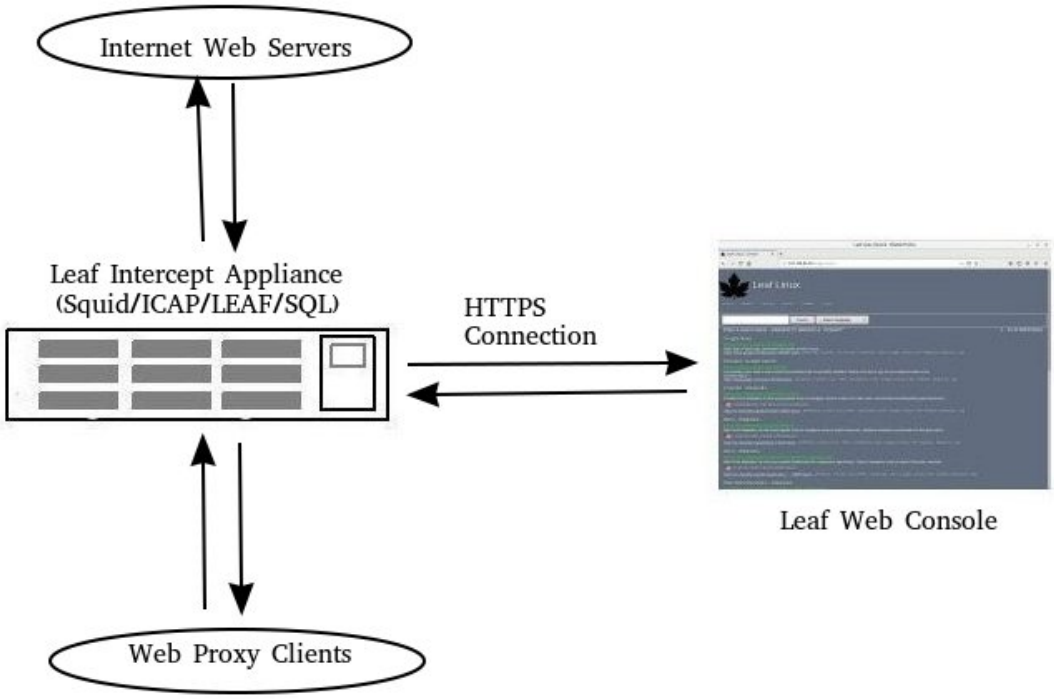
What is Leaf?



LEAF is implemented as an ICAP module which loads and runs under the C-ICAP Server. LEAF intercepts web pages from any ICAP capable proxy server, decrypts and decompresses the content, then forwards and stores it in an SQL database with full text search capability. LEAF can run on a local or remote SQL database and supports both IPv4 and IPv6 network protocols.

Leaf is a high performance intercept and storage technology which allows all web pages accessed through either a standard proxy server or a transparent proxy server to be stored at high data rates to back end databases. Leaf decrypts and decompresses all HTTP and HTTPS traffic, analyzes the traffic, then stores it into a background database with a full text search capability. The database supports a Google-like search engine interface for searching and reviewing captured pages to ensure corporate policy compliance, allowing system administrators to check for inappropriate usage of their networks, and quickly detects pages from adult themed websites, social networking websites, job search websites, and forum websites such as Wikipedia.

Leaf solves the customer problem of enabling full visibility of web site traffic on corporate networks for system administrators and also supports rapid searching of captured web content for verifying that corporate networks are being used in compliance with corporate network use policies.

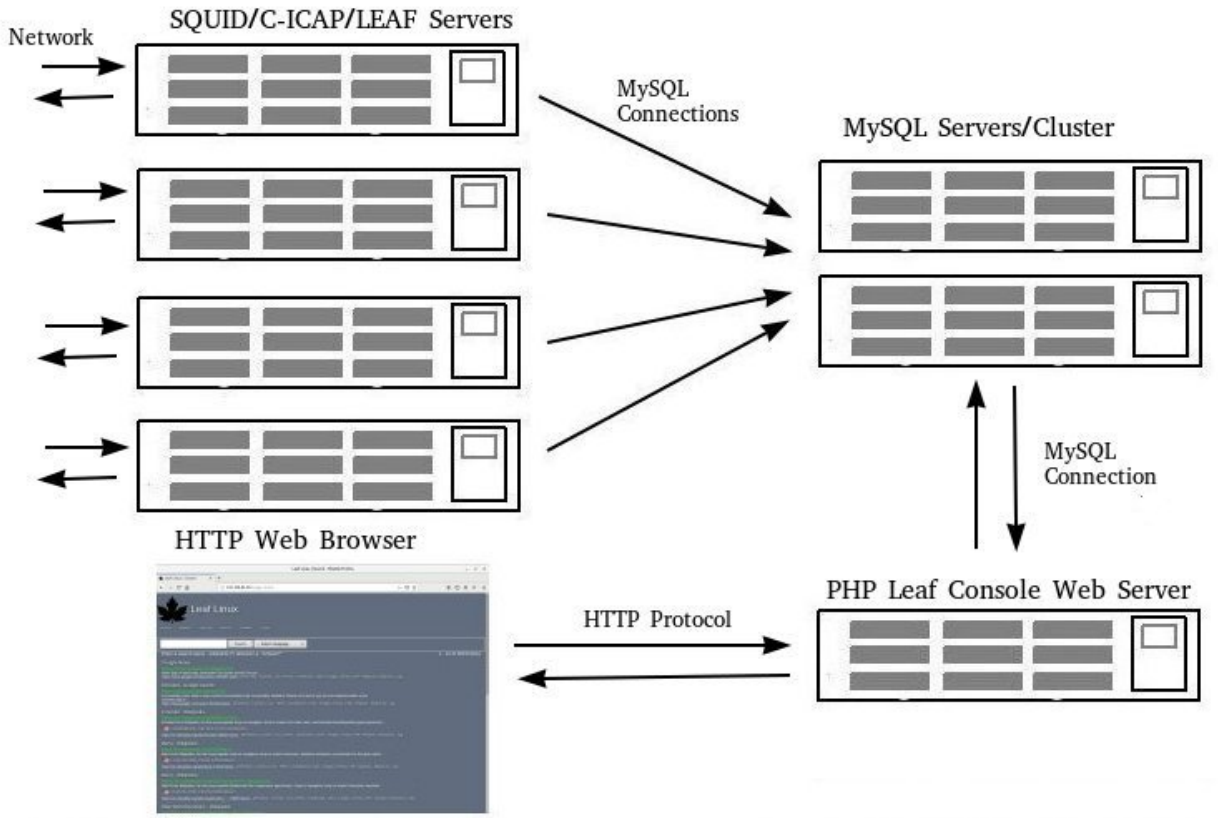


In the most simple and basic configuration, a Leaf Intercept Appliance is installed as a manual or transparent proxy server with C-ICAP, Leaf, MySQL, and the Leaf Web Server on a single high end appliance chassis. For remote office configurations, a single appliance can handle several hundred users. Larger installations may require multiple clustered appliances with the Proxy Server component installed in a separate appliance chassis from the C-ICAP server. The SQL Server and Leaf Web Console can also be segregated on unique systems and supports complex configurations with clustered database and appliance arrays feeding into a distributed database.

The interface used to achieve this is the same interface commonly used by proxy servers for url filtering and anti-virus web page scanning, and is implemented based on the ICAP standard (Internet Content Adaptation Protocol) and associated communication protocols. Virtually all commercial proxy servers, including SQUID and the Microsoft Proxy Server, fully support the ICAP protocol.

The ICAP architecture is typically implemented as a standalone server which receives ICAP messages from the host proxy server, allowing web pages to

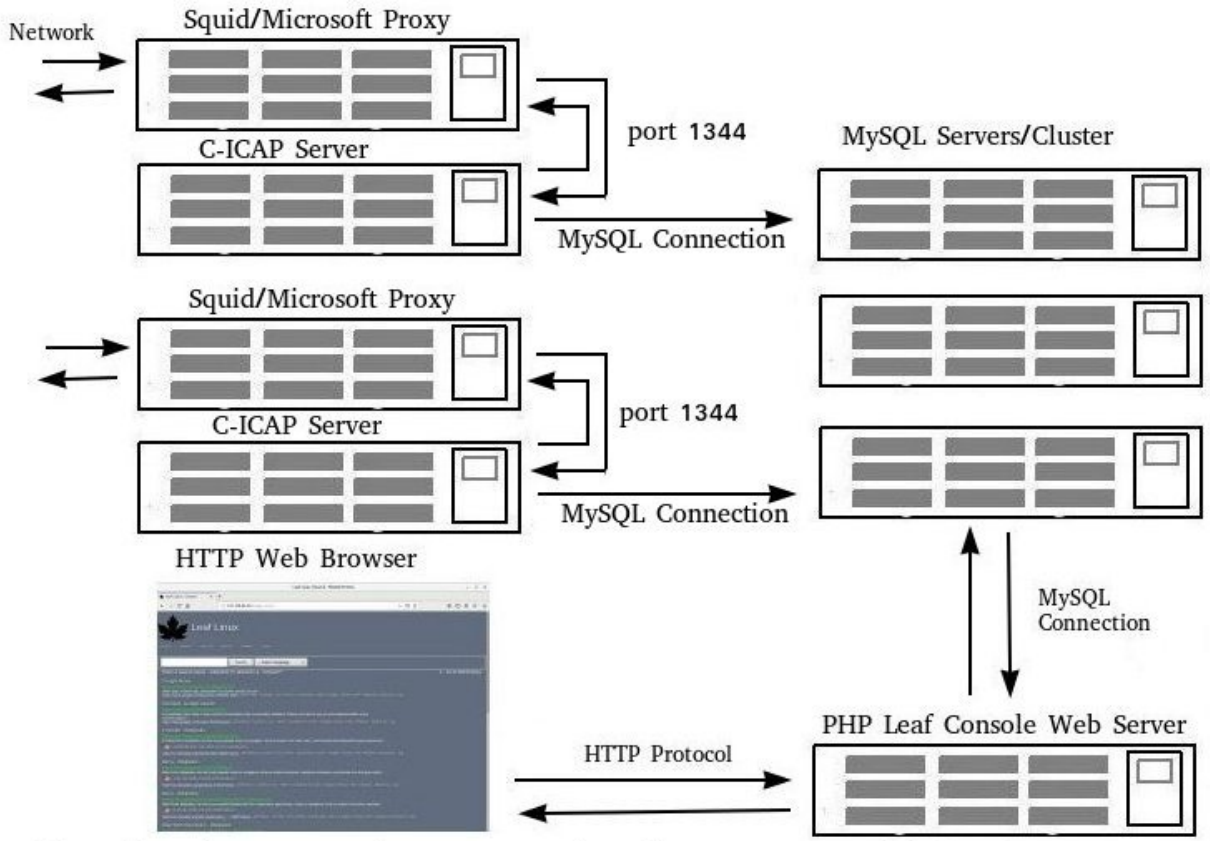
be scanned for software viruses and sending ICAP response messages and virus-free HTML content to proxy users. Leaf takes this a step further and decompresses received compressed web content, analyzes the content, then stores it into an SQL based data storage model.



In this configuration, multiple SQUID proxy servers are configured with C-ICAP and Leaf which are connected to backend MySQL database servers or a MySQL Cluster. The PHP Web console for viewing the captured web pages in this configuration is a standalone web server or client which also connects remotely to the MySQL servers. This particular configuration allows easy integration if a customer is already running SQUID proxy servers since C-ICAP can be installed on each Proxy Server.

The Leaf ICAP intercept module is proprietary and ‘C’ language based and is provided to the customer as a binary module and each module requires a license which is tied to the detected hardware on each customer system to prevent piracy and unauthorized copying. The Leaf binary modules are not open sourced. The base C-ICAP server is licensed under the LGPL (lesser GPL) which means that proprietary binary modules are fully supported and allowed to run time link against the C-ICAP server core without the requirement they be open sourced.

The Leaf search engine console for searching captured web pages is PHP based and the source code for the Web Console is included with Leaf, which provides customers the ability to modify and customize the search engine for each customer's unique requirements. This customer flexibility is further enhanced by allowing Leaf to run with either Linux or Windows based proxy servers and a range of SQL server architectures via ODBC modules.



This configuration separates the Proxy server from the ICAP server and deploys them on separate appliance chassis. This configuration effectively doubles the number of network sockets available to proxy users and should be used in deployments which support over 500 proxy users. This setup also allows any proxy server which supports the ICAP protocol to be integrated with a Leaf appliance. The network connection between the Proxy and ICAP servers should at a minumum be 1GB ethernet.

The base technology is currently implemented on Linux (Centos 7). The base technology comprises four distinct technologies which can be installed on a single appliance, or they can be distributed across several servers which can

be a mixture of Linux and Windows systems. At present, the C-ICAP server and modules is the only component which requires a Linux based server.

Because Proxy Server protocols, ICAP protocols, SQL protocols, and PHP web services are all fully distributed models, these four components can be distributed across multiple systems based on network topology and easily clustered, allowing multiple Leaf ICAP appliances or servers to communicate to a backend SQL database cluster or storage array by using commodity platform services.

This flexibility allows Leaf to be installed on most cloud based systems such as Azure and AWS as either a standalone ICAP interception server, or a combined appliance image with SQUID and C-ICAP attached to a back end database server. The PHP search engine console can also be installed on a standalone system with remote SQL access or it can be combined with the other three components on a single appliance. Providing the customer with the PHP source code is an attractive option and allows customers the freedom to modify the search engine to provide specific reports and data views based on each customers particular requirements. Providing the PHP source code for the search console also creates opportunities for consulting services and custom console development for customers who are willing to pay for such services rather than do the development work in-house, since the customer essentially owns their own unique version and enhancements of the search engine.

The ability of Leaf to make use of commodity SQL database platforms is an attractive approach for most customers who probably already have a preexisting database infrastructure for their corporate data. This approach means that minimal integration overhead is required to implement Leaf in existing infrastructures and cloud based systems. If customers are already using SQUID or another proxy server, Leaf can be integrated as a standalone ICAP server or into any preexisting C-ICAP based installation.

Modern network appliances have evolved to the point they provide sufficient disk and network bandwidth to support SQL based stream to disk capability. Testing has demonstrated that even a modest desktop system can meet the minimum performance requirements for HTML stream to disk for a small to medium sized office configuration.

Deep Packet Capture vs. Proxy Interception

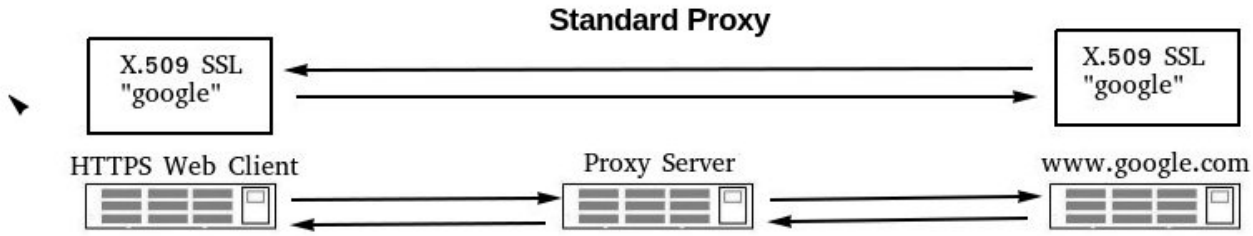
During the late 1990's and early 2000's, networking solutions available to customers for achieving network visibility of website traffic were for the most part reliant on deep packet capture technologies which captured all network traffic by streaming captured network packets to disk, then employed some form of reconstruction console to reassemble and display web pages, ftp, P2P, and other forms of network traffic. Over time, the internet began to evolve towards wholesale adoption of TLS and HTTPS protocols, with virtually all web based traffic being encrypted and utilizing compression techniques as of 2020.

Because of the way that public/private key encryption models function, most deep packet capture technologies became unable to efficiently process or decrypt HTTPS traffic without access to the secret key for a particular server. Although deep packet capture technologies do enable a complete view of who is on your network and what nodes they are communicating with, these technologies do not enable the ability to view the actual content being sent over the network. SSL/TLS Proxy interception (also called SSL Bumping) began to appear in network proxy servers such as SQUID in order to enable web pages to be decrypted to detect viruses and certain types of prohibited urls or web content.

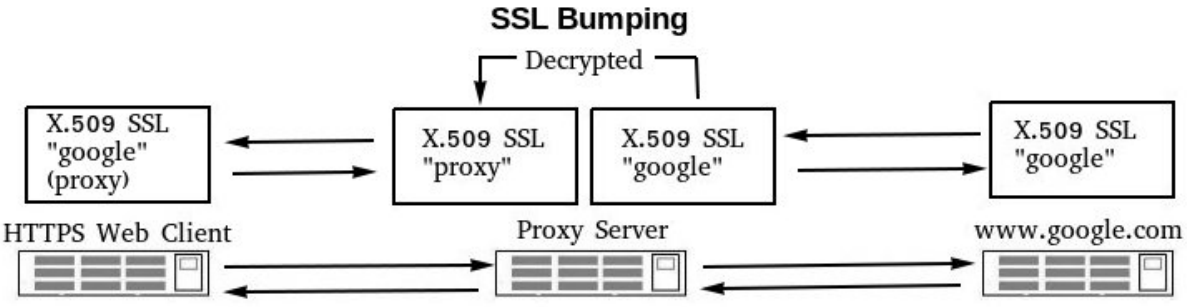
Modern proxy servers all support SSL bumping in some form for proxy clients, and both standard proxy and transparent proxy modes are fully supported currently by SQUID 4.10.

What is SSL Bumping?

SSL bumping is service provided by a proxy server which decrypts and encrypts HTTP transactions between the proxy client and the remote web server, allowing HTML content to be decrypted and reviewed. Internet Web Servers which are configured to support HTTPS connections are configured with an SSL certificate which is used to establish a secure connection between the server and clients. When a proxy client accesses a remote server via the proxy server, it normally uses the remote servers SSL certificate to establish a secure connection to it.



When a proxy server is configured in standard mode, it passes remote server X.509 TLS certificates all the way to the web client for establishment of an SSL/TLS connection. In this mode, the proxy server functions as an HTTP transaction router, and caches then routes remote SSL credentials. The originating domain is contained in the SSL/TLS data exchanged between the web client and remote server.



When the proxy server is configured to employ SSL Bumping, the proxy server establishes a secure connection to the remote web server with the remote server X.509 certificate, then on the web client side of the proxy uses an intermediate certificate to establish a TLS/SSL connection between the web client and the proxy server. The proxy server essentially "forges" the proxy side certificate to mimic the certificate from the remote web server certificate. This allows the data that flows between these two independent TLS/SSL connection tunnels to be decrypted and passed to ICAP services for URL checking, language translation, anti-virus scanning, and web interception.

With SSL Bumping enabled, a second SSL certificate is hosted on the proxy server which is used to establish an SSL connection between the proxy server and the proxy client, and the proxy server then establishes an independent SSL connection to the remote web server with the remote servers SSL certificate. This creates two distinct and independent SSL connections, one between the proxy client and the proxy server, and another between the proxy server and the remote web server. Data flows between the remote web server and the proxy server are then decrypted and passed to an ICAP server for analysis before being re-encrypted and sent to the proxy client. ICAP analysis services can include virus detection, url blocking, language translation, or interception of the web pages for capture.

From the proxy clients perspective, the certificate presented to it appears to belong to the remote server. This is because the proxy server essentially "forges" the remote web servers identity into the SSL certificate hosted on the proxy server before sending it to the proxy client. SSL Bumping works well for almost all modern websites, and in almost all situations cannot be detected by either the web browser or proxy client applications. Most customers who use ClamAV anti-virus detection with SQUID already have their own internal SSL certificates installed they use for this purpose, and may be already using SSL Bumping to support anti-virus software.

HTTP Compression Support

When a web browser sends an HTTP request to a remote web server, modern web browsers attach a message header to the HTTP request specifying which compression/decompression models are supported by the web browser. Most web traffic is both encrypted and compressed by the web server in order to improve performance and decrease bandwidth usage. By way of example, Google News main pages are over 3MB in size and are compressed to a size of 400K before being transmitted over the internet. Leaf supports all current compression/decompression models used by web servers on the internet. The current supported compression/decompression

models currently in use on the internet are deflate/inflate, gzip/gunzip, bzip/bunzip, bzip2/bunzip2, and brotli (Google).

Implementation

The current Leaf server appliance was developed on Red Hat Enterprise Server version 7/CentOS 7. The appliance uses a Centos 7 base operating system with the additional Leaf Server RPMs (Red Hat Packages/Modules). The PHP web console can run on any web server that supports PHP, including Linux and Windows Web servers. The C-ICAP server and Leaf ICAP Intercept modules currently run on Centos7, however the C-ICAP server can be recompiled to run on Windows.

C-ICAP can also be configured to run on a different appliance from the Proxy server in order to increase available port sockets for large numbers of web clients. In deployments which host large numbers of web proxy clients per proxy server, it is strongly recommended to deploy the proxy server on a separate appliance chassis from the C-ICAP server, and that both be configured to use as a minimum a gigabit network backbone between the proxy server and the C-ICAP server, since this will double the available port sockets for client and server usage.

In configurations which support a small number of proxy clients, the proxy server and ICAP server can be combined along with MySQL and the Leaf Console web server on a single appliance chassis.

The Leaf Intercept module is proprietary and closed source (c-icap-leaf). The Squid Proxy Server v4.10 is released under the GPLv2. The C-ICAP source base is released under the Lesser GPL. The Lesser GPL allows proprietary closed source modules such as the Leaf intercept module to link to it. The Leaf appliance build listing consists of the following binary RPM modules/Images which can be installed with the rpm package manager utility (rpm or yum). Using a Yum repository allows end users to download

and install updated modules or new features as required in an automated manner:

GPL v2

CentOS 7 DVD Image (x86_64 build 1908)
squid4-4.10-1.el7.leaf.x86_64.rpm (Squid 4.10 Core)*
squid4-debuginfo-4.10-1.el7.leaf.x86_64.rpm
**Note: There are no changes to the Squid base source code required to support Leaf. The only changes are to the rpm build file (squid4.spec) to enable compile time build options to enable icap client support and SSL Bumping to decrypt web sessions.*

Lesser GPL v2.1

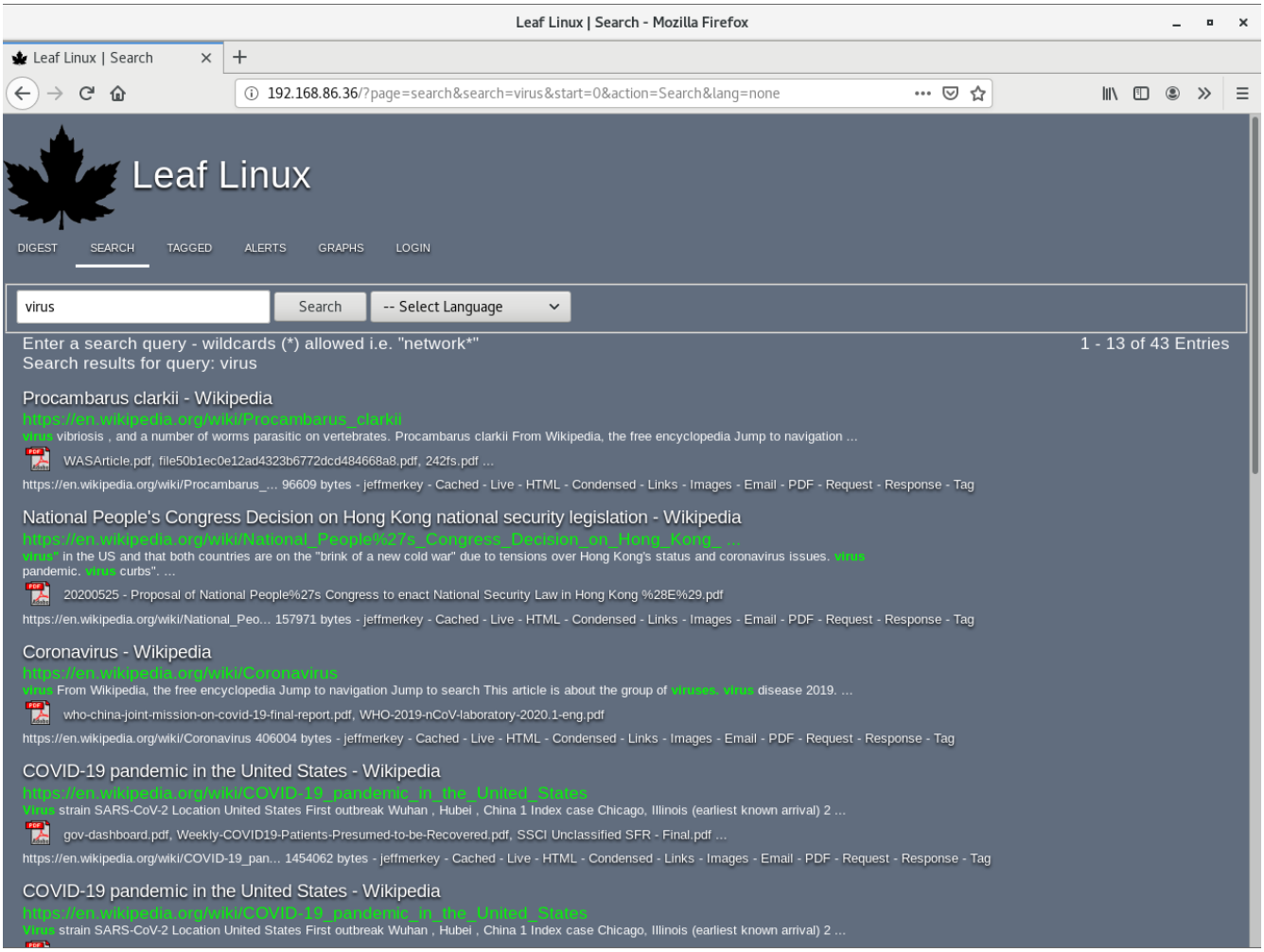
c-icap-0.5.6-1.el7.leaf.x86_64.rpm (C-ICAP Core)*
c-icap-debuginfo-0.5.6-1.el7.leaf.x86_64.rpm
c-icap-devel-0.5.6-1.el7.leaf.x86_64.rpm
c-icap-ldap-0.5.6-1.el7.leaf.x86_64.rpm
c-icap-libs-0.5.6-1.el7.leaf.x86_64.rpm
c-icap-perl-0.5.6-1.el7.leaf.x86_64.rpm
c-icap-progs-0.5.6-1.el7.leaf.x86_64.rpm
**Note: There are two bug fixes to the base ICAP code and an additional posix shared memory segment allocated. These fixes were to correct two separate memory leaks detected during load testing and also enable shared memory support for ICAP server statistics for the Leaf Server Monitor.*

Closed Source/Proprietary

c-icap-leaf-0.5.6-1.el7.leaf.x86_64.rpm (Leaf Intercept Module)
leafcon-1.3-el7.leaf.el7.noarch.rpm (Leaf PHP Web Console)
leafmon-1.3-el7.leaf.el7.x86_64.rpm (Leaf Server Monitor)

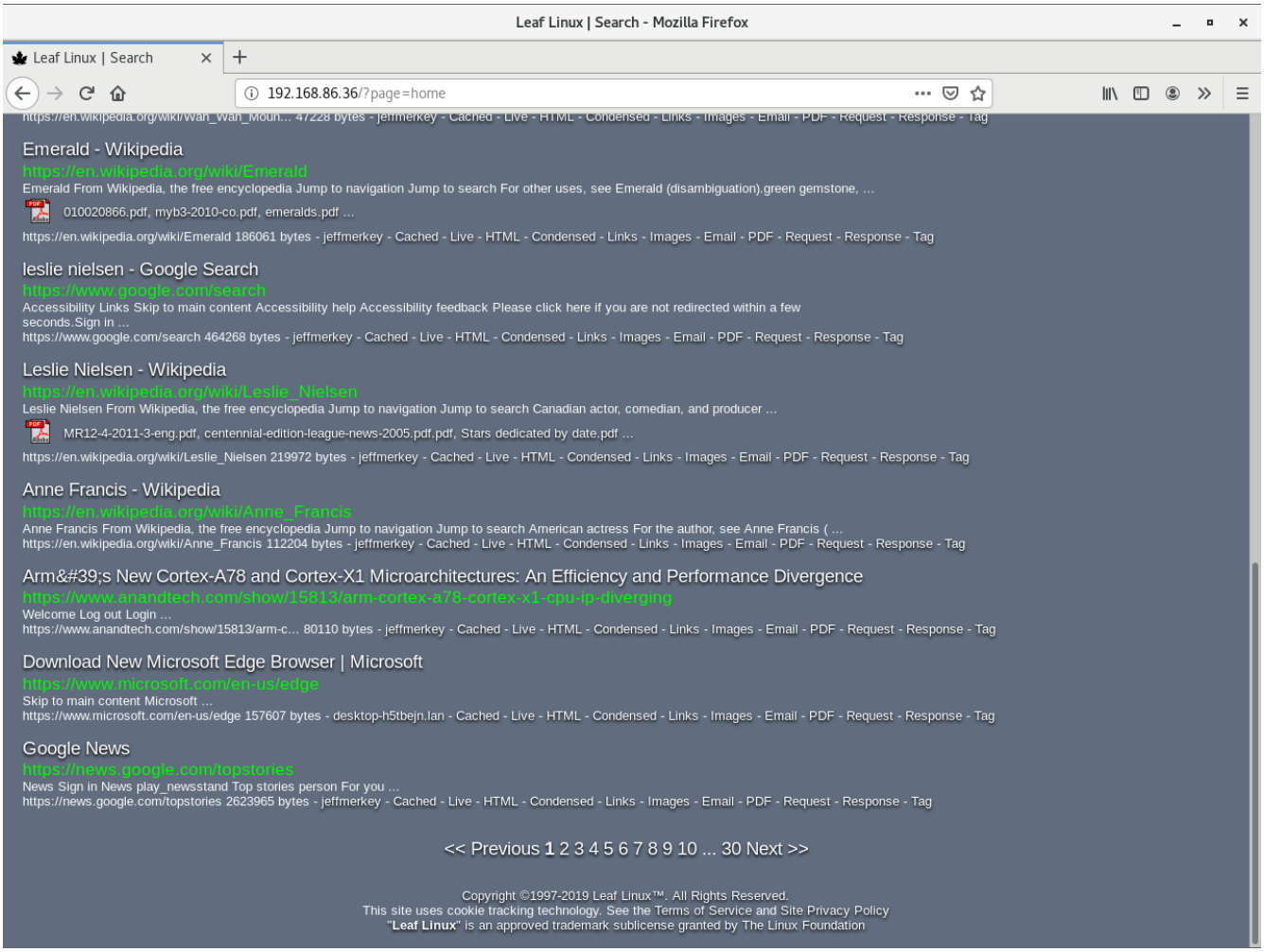
Leaf Search Console

The Leaf search console is a PHP based standalone web server which can access, scan, and reconstruct web pages for viewing by a network administrator after they have been captured and written into an SQL database by the Leaf intercept module. The web console interface is PHP/MySQL based and will function with any HTTP web browser. The PHP base code can be easily modified or enhanced based on customer requirements.



Leaf Web Console top of page search panel. The search interface is google-like and can be configured to use a variety of different style sheets to change the look and feel of the web console.

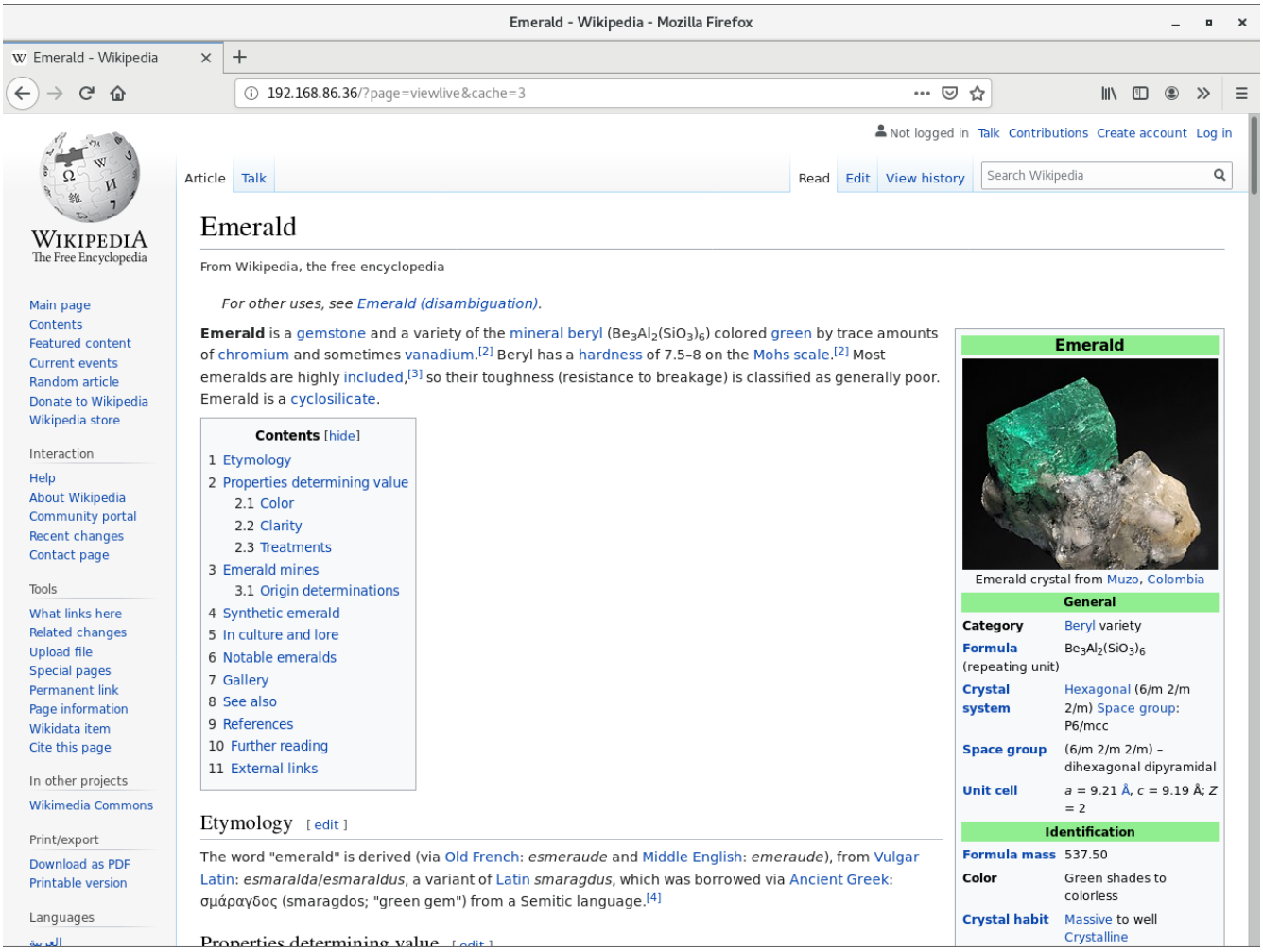
The Leaf web server can scan, load, and then run captured pages “live” on the internet from the server cache as well as providing a static view of a cached web page. The console also will display the raw HTML page text (HTML), fulltext search extracts (condensed), IP Address information for monitored proxy clients, DNS names, proxy user accounts, embedded page objects such as PDF files, as well as a list of all links and images referenced by a particular page, along with HTTP request and response headers for a particular page.



Leaf Console bottom of page search panel. The bottom search panel provides pagination support similar to a google-like search panel

The web console provides a google-like search engine interface for searching captured pages and provides fulltext search capability of captured web

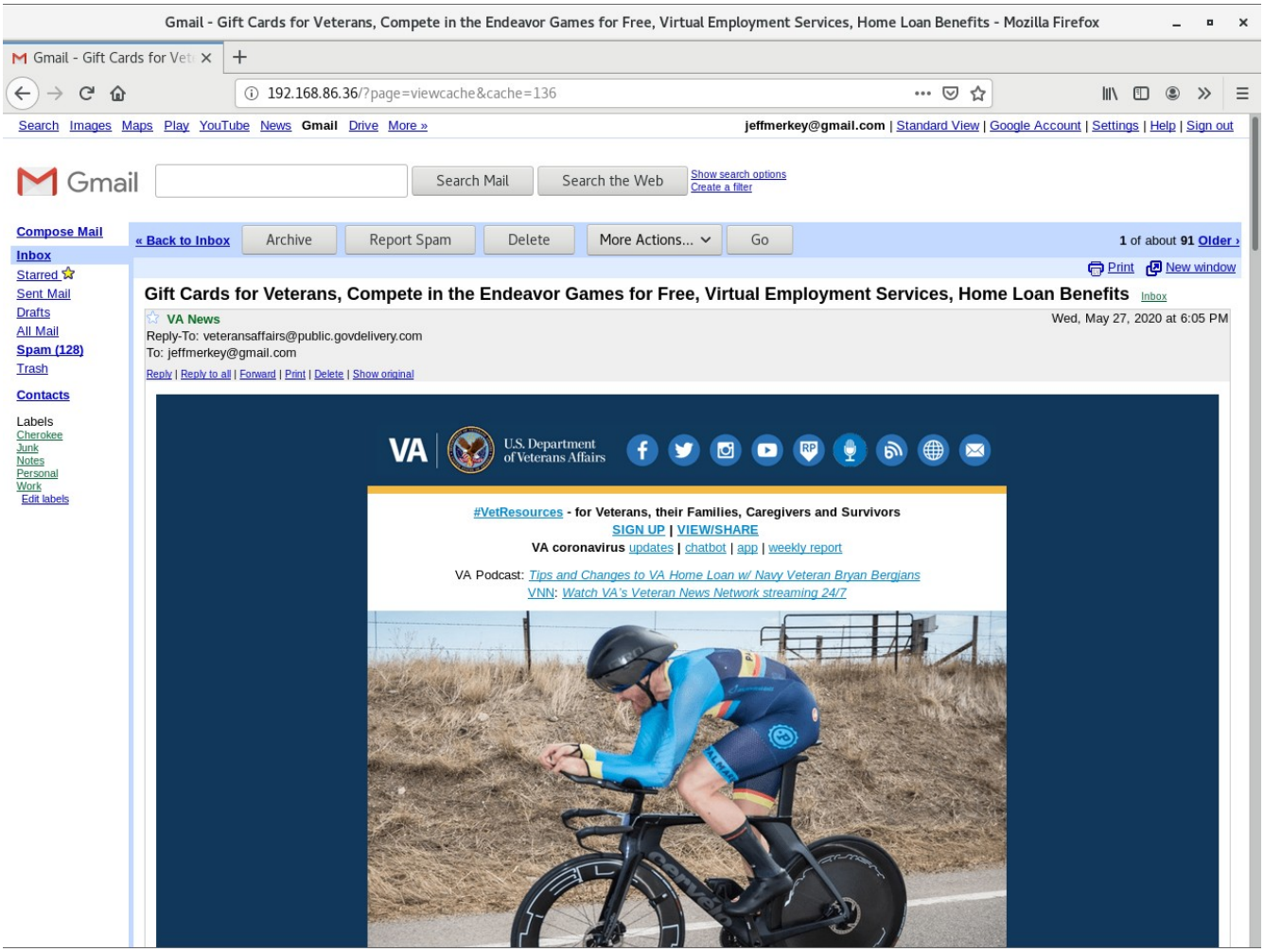
content. Captured pages older than 30 days can be configured to be automatically archived into another database, or they can be output as .sql files or CSV files (comma delimited files), which will increase database search speeds of the most recent data. If multiple Leaf capture appliances are deployed together, each appliance can be configured to use a unique database table name or database server name.



Leaf Console reconstructed web page for “Emerald” using the “live” view. This view loads the web page, performs href, img, and relative url fixup to point to the originating site, then runs the page scripts as live.

These back-end databases can then be configured to record web pages in parallel SQL tables to boost performance, or they can also be configured to funnel into a single high availability database or database cluster. The

console can also be enabled to provide combined views of captured data across multiple tables or databases.



Leaf Console reconstructed web page from a captured Gmail session "cached" view.
This view loads the web page from static storage. This viewing mode is useful for sites that require a login such as gmail. The cached page can be viewed independent of the live server in this mode.

The Web console also supports page tagging, and individual pages can be tagged which will allow them to be collected and displayed in a condensed console view. This allows network administrators to tag pages which may be of interest for later review. By default, the Leaf database module only stores pages of the text/html MIME type. Leaf can be configured to record all non text/html pages as well, although doing so can utilize considerable database storage and negatively impact end user web browser performance when

going through a proxy server since all of this content has to be sent in total to the ICAP server instead of only text/html pages.

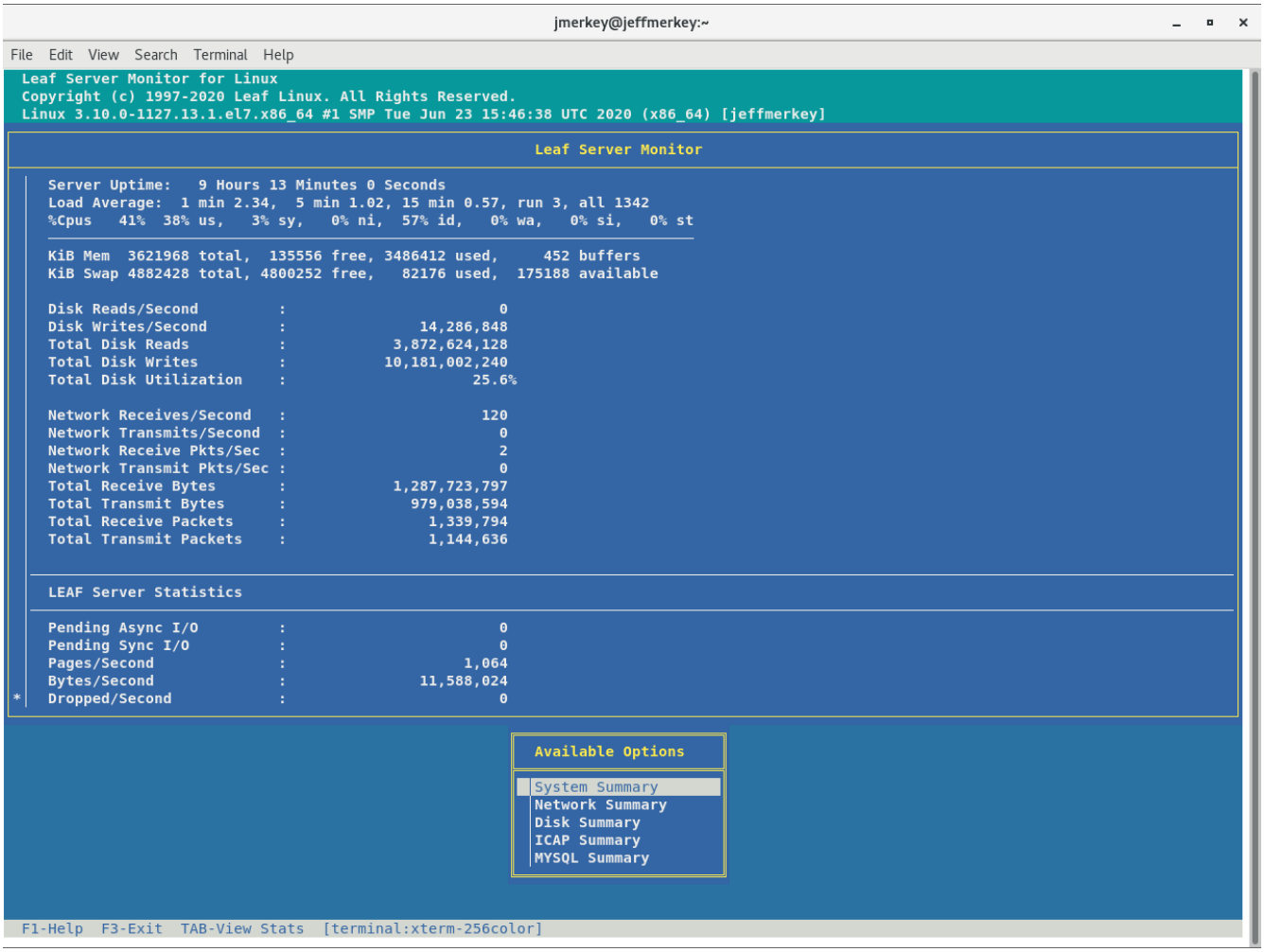
Since the majority of non text/html pages contain mostly javascripts, json objects, or video content, these files are not required in order to run a captured page as live. When the Web console runs a captured page as "live" from it's page cache, the PHP code analyzes all url link fields, images, and scripts and performs a url fixup which allows cached pages to be fully reconstructed in the Web Console web browser. The web console also provides the original url which can simply be clicked on to visit the website the page originated from with the same functionality as google's web interface.

ICAP servers by default are configured to receive 1024 bytes of preview data for each web object sent from a proxy server for review by ICAP services and modules. Before ICAP signals the proxy server to send the entire object to the ICAP server, ICAP modules have the option of signaling the proxy server with a 204 return code which informs the proxy server to release the end user web client request and send the web data to the proxy client without ICAP module review or receipt. For text/html pages, by default Leaf tells the proxy server to send the entire page to the ICAP server for processing and returns a 204 Status for any page preview which is not text/html content, which dramatically improves ICAP server performance.

The Web console will also display a date ordered digest view of daily web traffic which can be organized and searched by IP address, proxy username, DNS hostnames, and for multi-appliance configurations, a local view of each unique capture appliance web pages. ICAP provides LDAP library bindings and LDAP support can be easily integrated into the Leaf Intercept server and Leaf Web Console for resolving user names from IP addresses or DNS names. The Leaf Intercept module also stores SQUID Proxy username and login information from the proxy server allowing proxy clients to be tracked.

Leaf Server Monitor

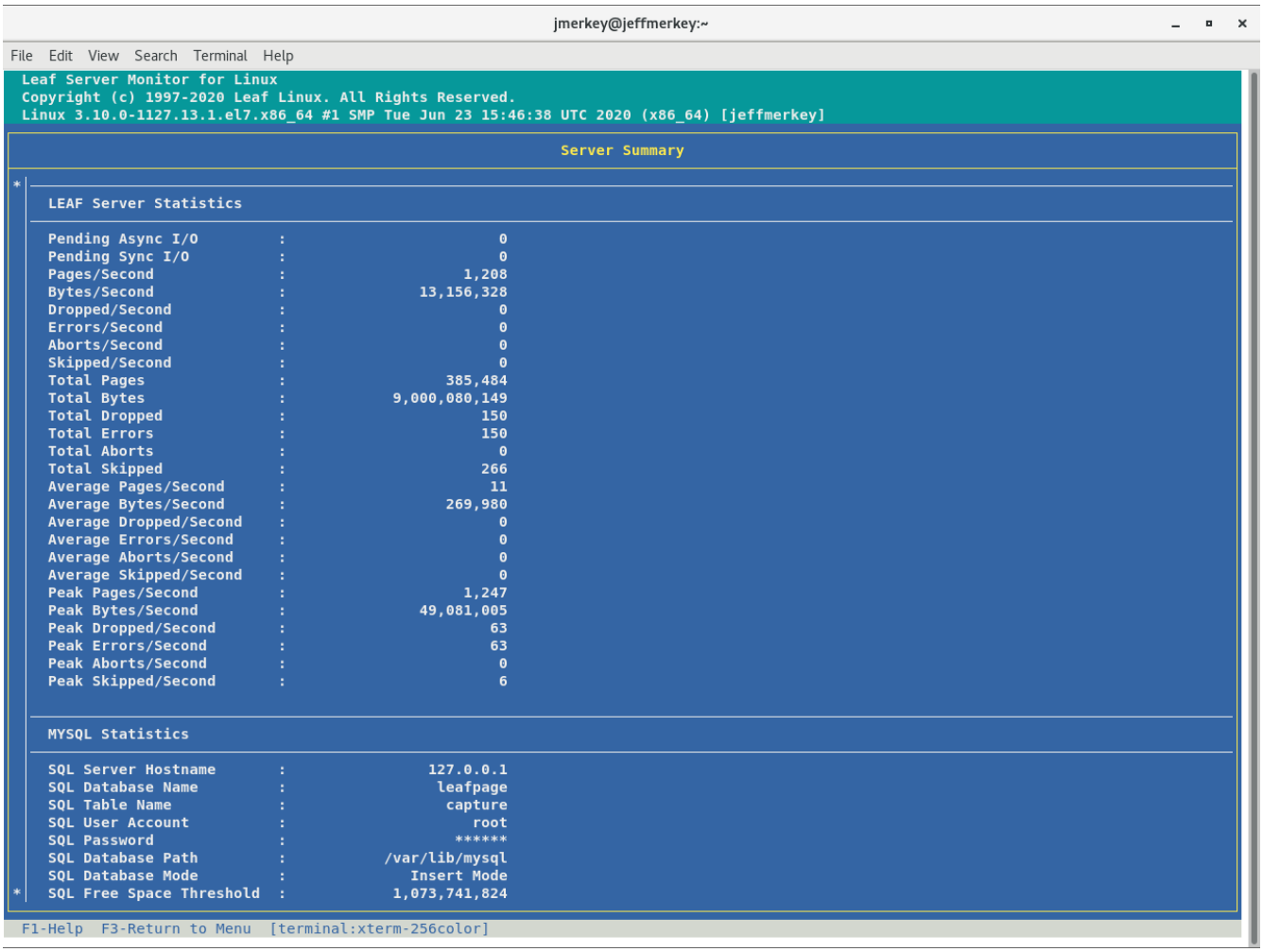
The Leaf Server Monitor (leafmon) is a Unix terminal based monitoring program that allows system administrators to monitor internal operating server statistics (such as disk I/O stats, processor stats, memory stats, networking stats), C-ICAP server statistics, SQL database usage and storage statistics, and Leaf Intercept Module Statistics.



Leaf Server monitor main menu (leafmon). The utility displays total disk and network throughput, processor utilization, memory usage, Leaf Intercept Server statistics, C-ICAP Server statistics, and MySQL statistics.

The Leaf monitoring program is written under Linux ncurses. Because the program is ncurses based, it will run across all ncurses supported terminals

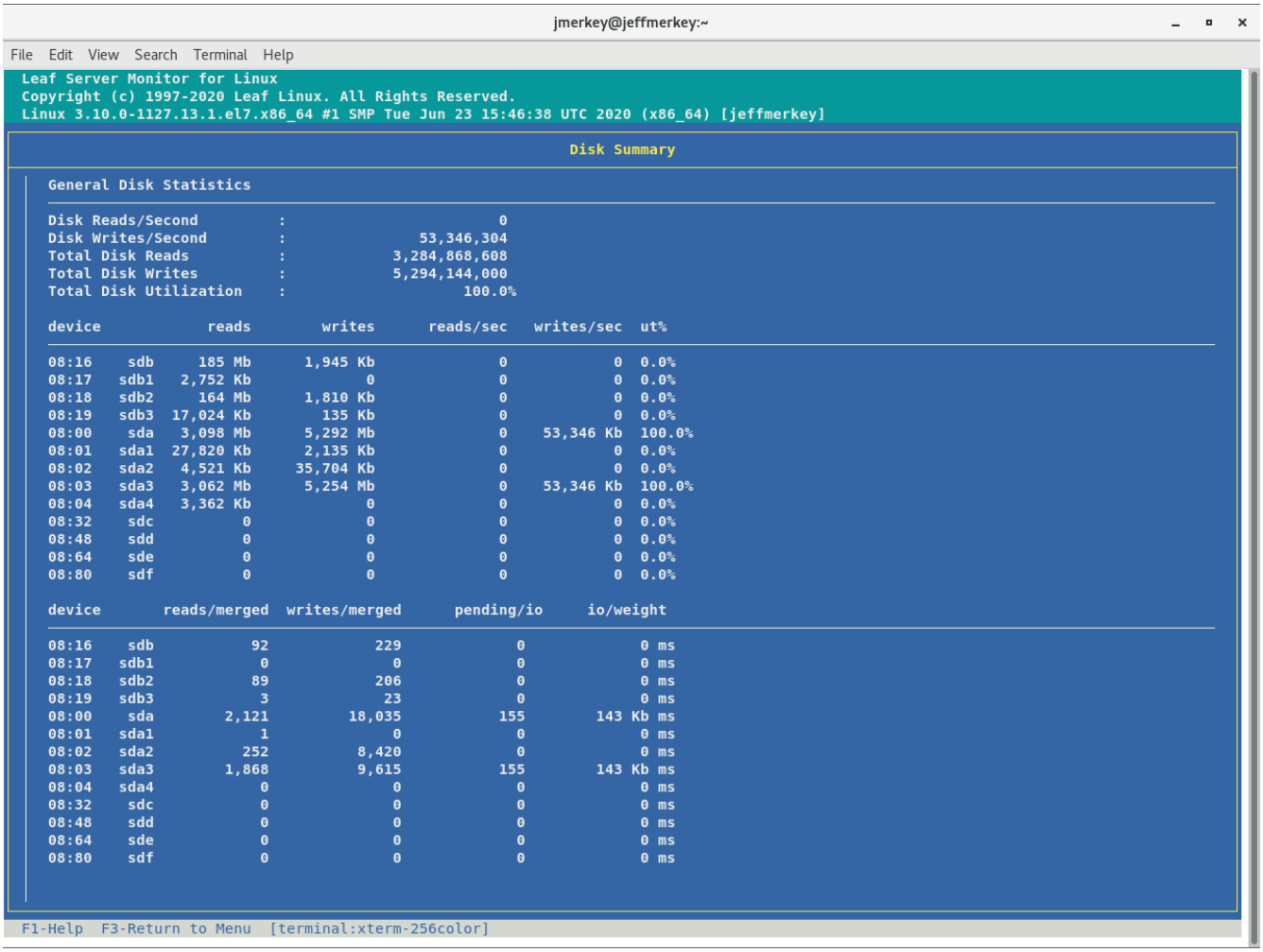
and terminal type emulations. This allows remote administrators to invoke the Leaf monitor via a BASH shell or secure shell (ssh) session as well as from hardware based vt100 terminal types and above.



Leaf Server monitor server summary for the Leaf Intercept module. This session reports that the Leaf Server is writing 1,208 html pages per second into the MySQL database with 1,247 peak html pages per second. The Leaf Server also reports it has written a total of 385,484 html pages to the MySQL database.

The Leaf monitor allows visibility into system performance and provides Leaf Intercept module statistics such as pages per second, total pages stored, SQL database storage usage, Leaf configuration and operation data, and detailed ICAP server statistics. It also displays disk utilization and disk reads and writes per second and network sends and receives per second.

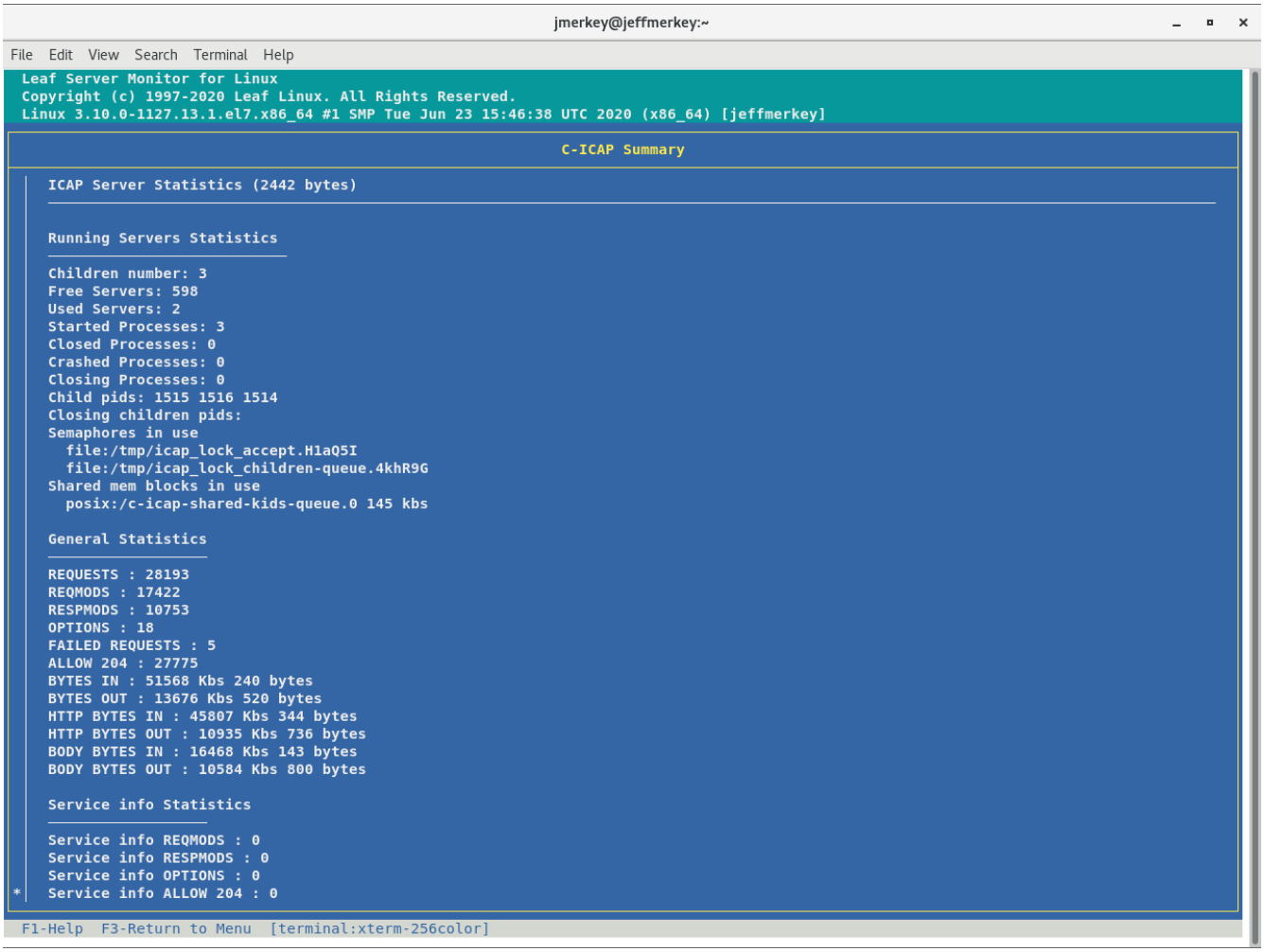
The Leaf monitor is extremely useful for providing visibility of performance and resource usage of a real time ICAP Server with active Leaf html interception in test environments and customer deployments. The leafmon tool enables rapid resolution of performance issues, tuning, or configuration issues.



Leaf Server Monitor detailed disk statistics panel. This panel lists all attached local and remote hard disk storage and live I/O statistics along with disk utilization. It also reports merged read and write requests and pending I/O.

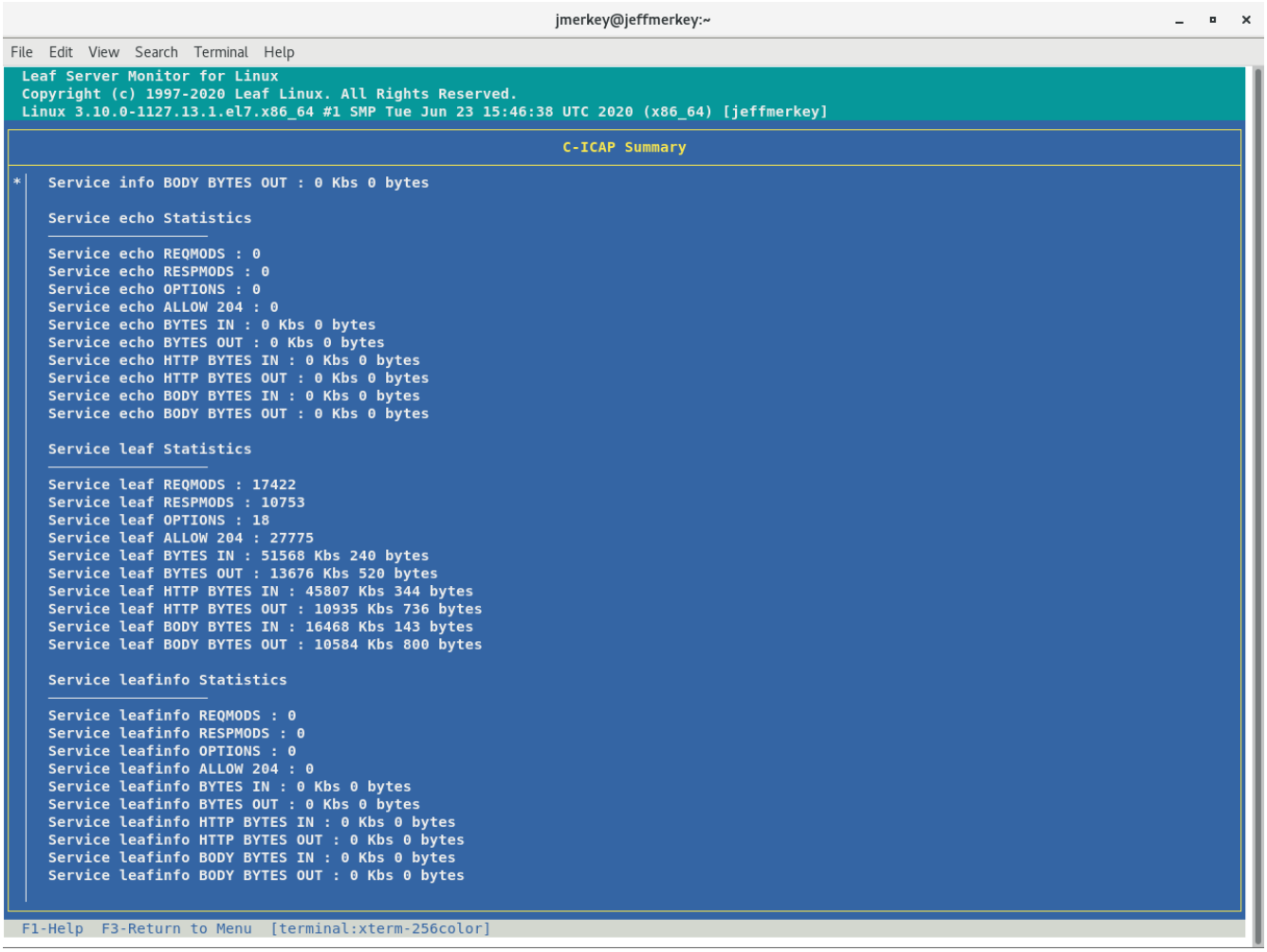
The Leaf Server Monitor also provides detailed statistics for the C-ICAP server including number of active servers, active processes, threads per process, total pages processed, total pages which were skipped with a 204 response (released and not processed by ICAP server), active C-ICAP

modules, and C-ICAP module statistics. REQMODS and RESPMODS are also tracked (proxy HTTP requests and remote web server HTTP responses). Administrators can view live ICAP sessions and session data in order to monitor C-ICAP performance, throughput, and network utilization.



Leaf Server Monitor C-ICAP summary panel. This panel displays real-time performance and transactional statistics for the main server core as well as for C-ICAP binary modules.

It is important to note that C-ICAP receives and processes both HTTP requests sent to a remote web server by the proxy server (REQMODS) and responses received from the remote server through the proxy server (RESPMODS). “ALLOW 204” is the number of pages the ICAP server told the proxy server it was skipping a proxy request (page was not text/html MIME type).

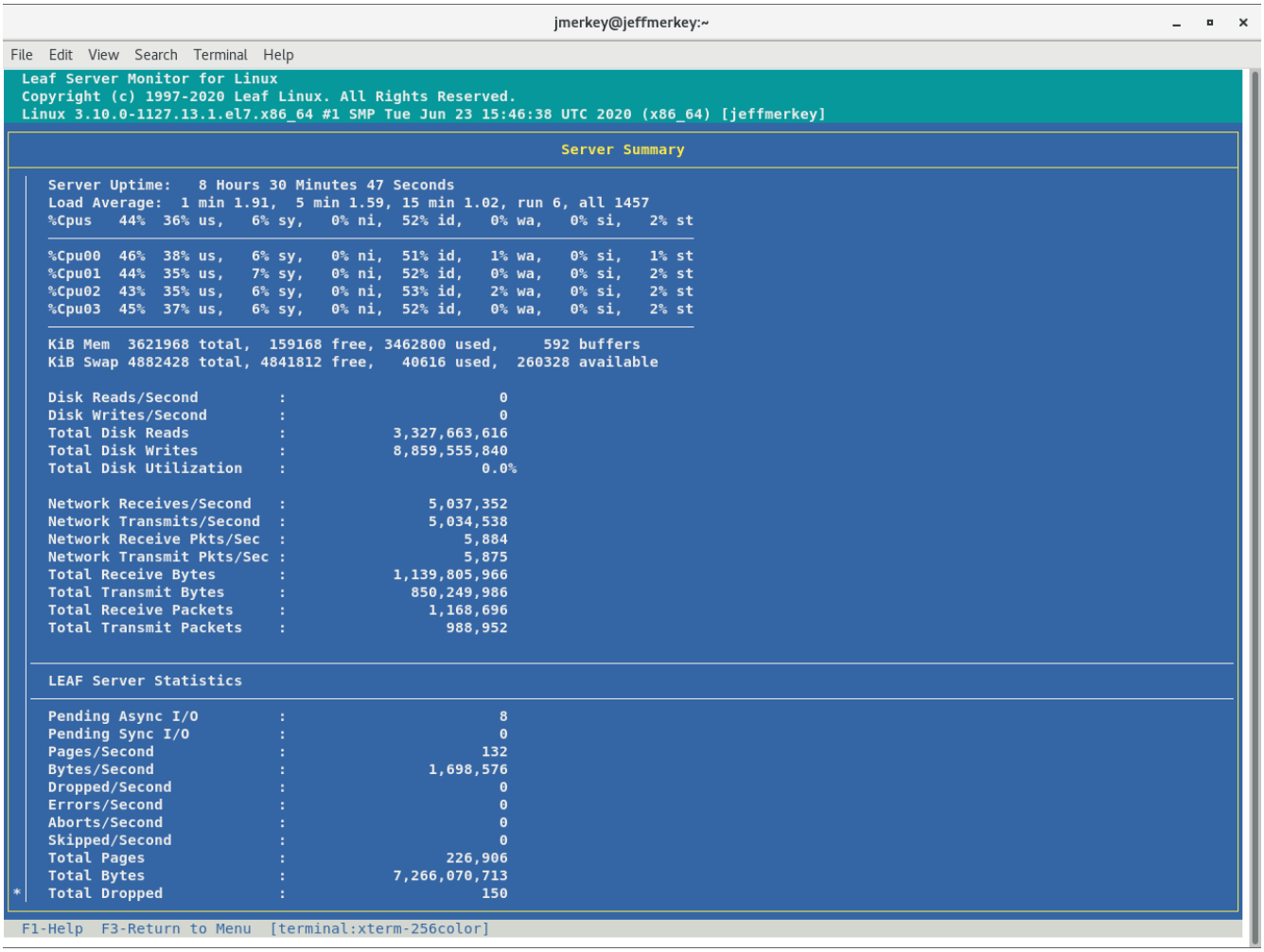


Leaf Server Monitor C-ICAP summary for the Leaf and Leafinfo C-ICAP binary modules. The monitor reports requests, responses, 204 status (skipped), along with bytes in and out for HTTP headers and HTML body data.

The Leaf Server Monitor provides five server-wide summary panels which report real-time statistics:

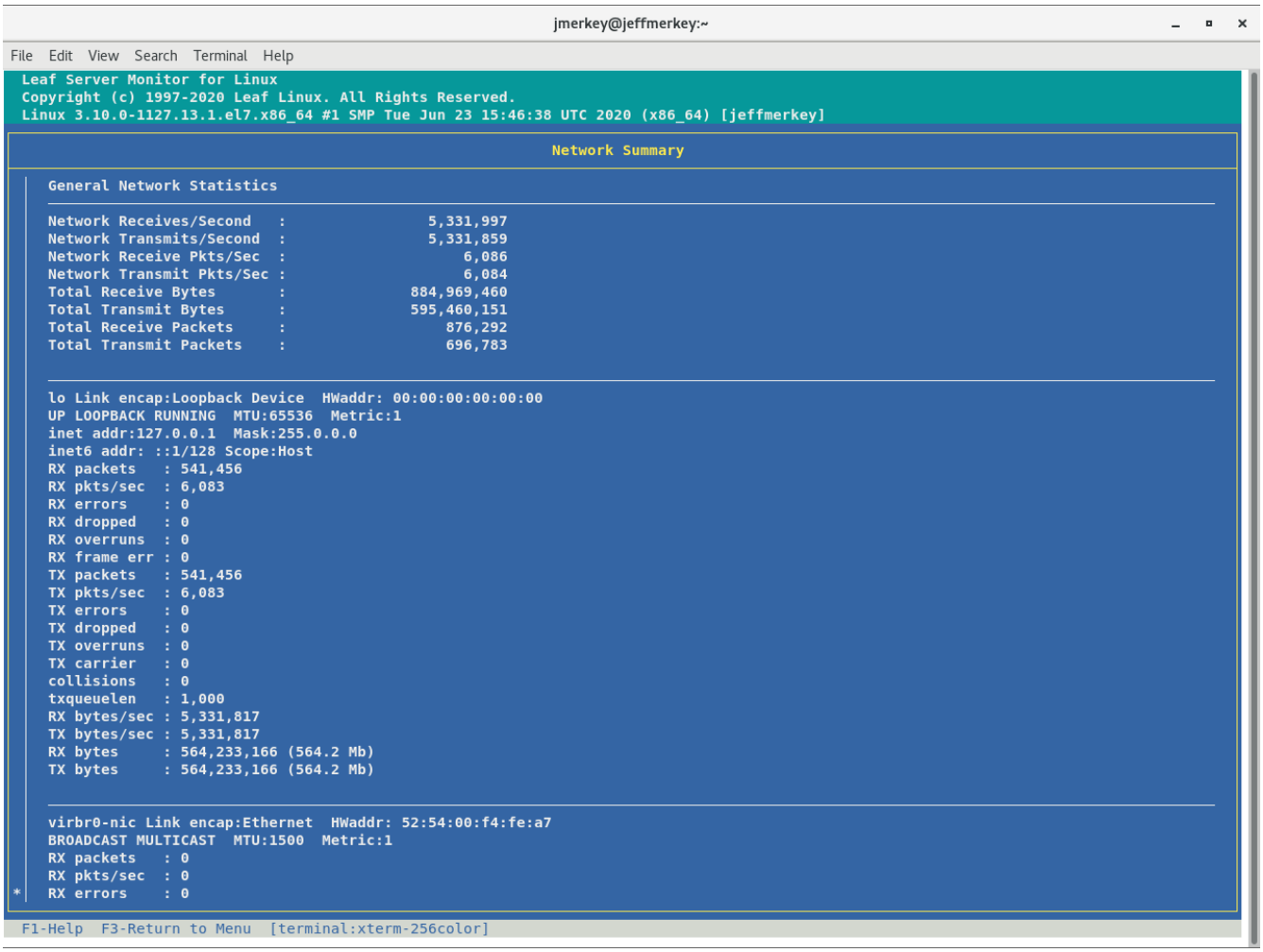
1. System Summary
2. Network Summary
3. Disk Summary
4. C-ICAP Summary
5. MYSQL Summary

1. System Summary – provides detailed system level statistics such as server up time, system load averages, processor utilization, memory usage, process states, total disk read/writes, network send/receives, Leaf Intercept summary statistics, MYSQL summary statistics and configuration.



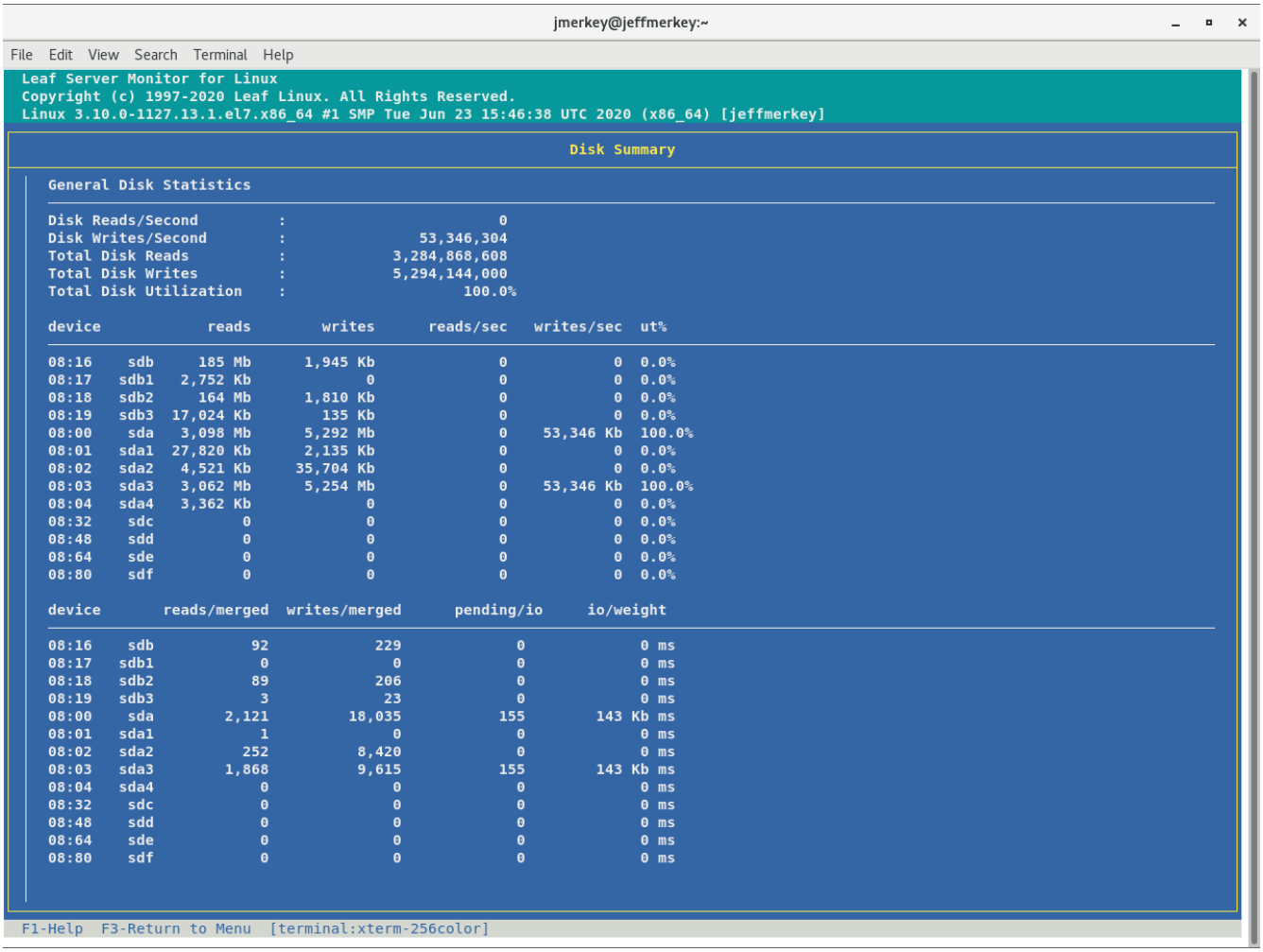
Leaf Server Monitor System Summary (expanded view)

2. Network Summary – provides detailed network adapter statistics, total send/receives, hardware adapter statistics, and a listing of all active and detected network adapters.



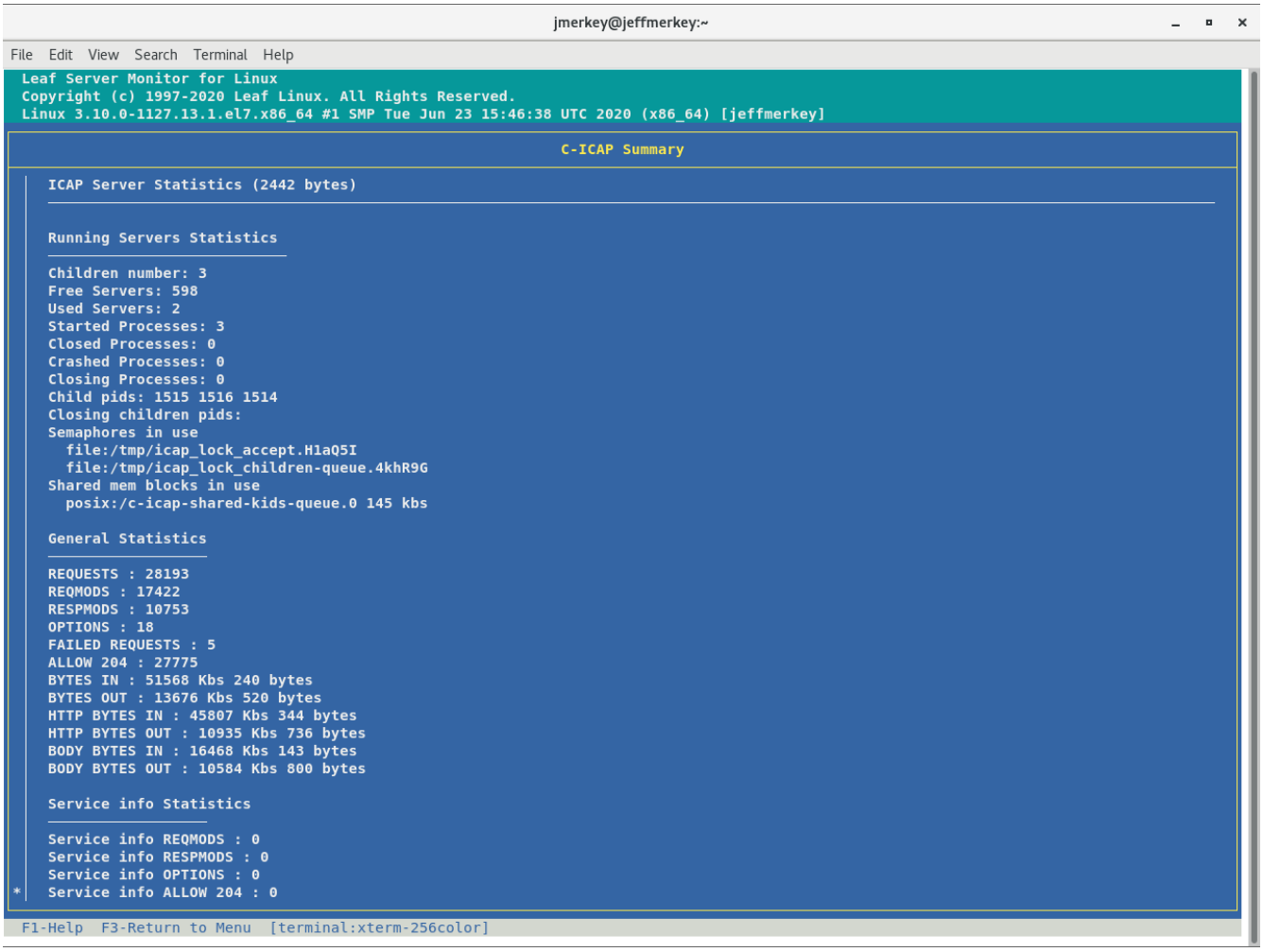
Leaf Server Monitor Network Summary (expanded view)

3. Disk Summary – provides detailed view of local and remote disk storage, disk utilization, disk read/writes per second, pending I/O requests, elevator and coalesced disk I/O requests (read and write merges), and I/O profiling of all disk devices as well as partition based views of I/O operations and data rates per active disk partitions.



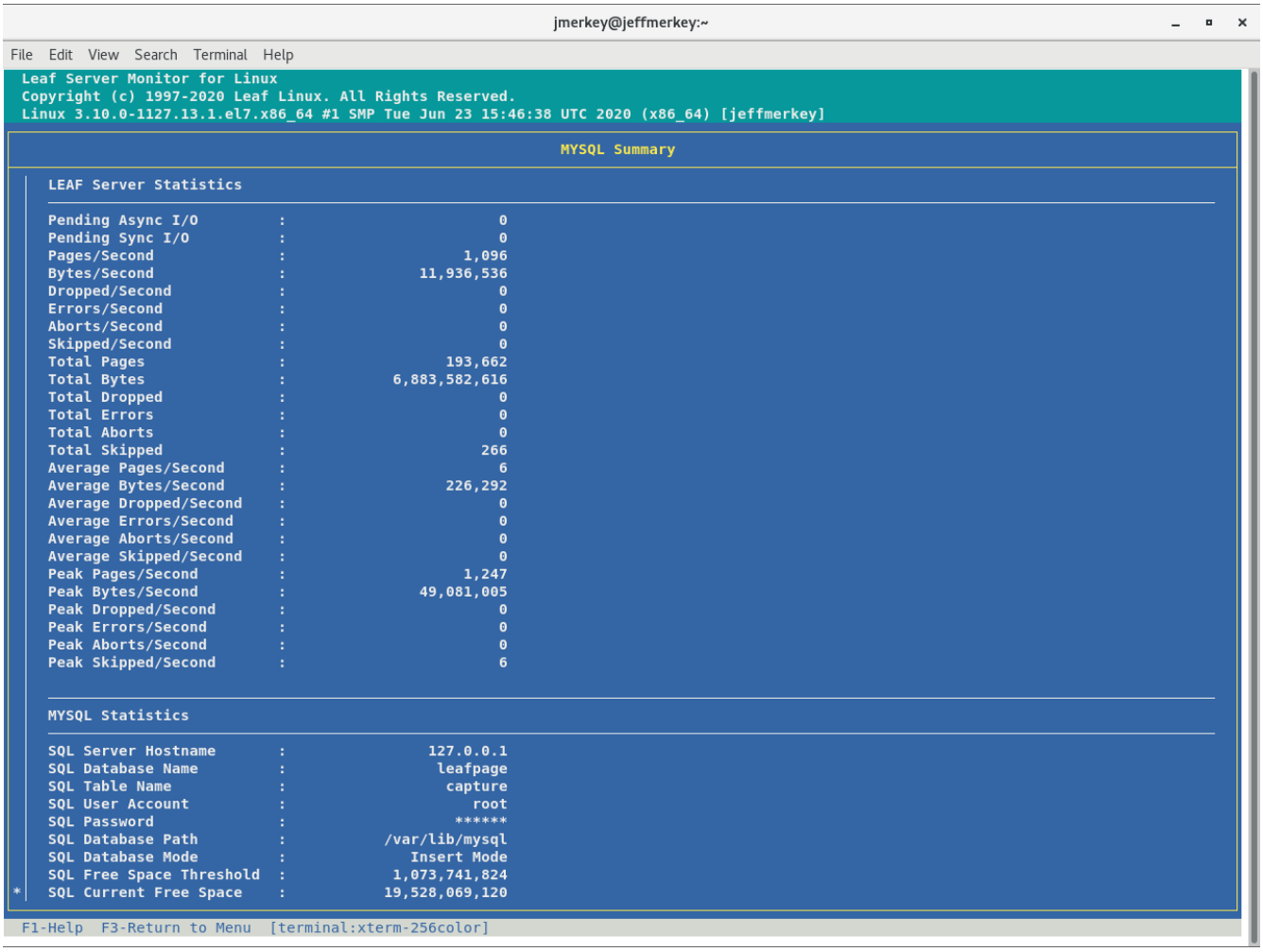
Leaf Server Monitor Disk Summary (expanded view)

4. ICAP Summary – provides detailed view of all C-ICAP server transactions, including ICAP requests per second, live ICAP server processes, HTTP bytes both in and out, detailed breakdown of loaded C-ICAP module statistics, errors, skipped requests (allow 204 responses), and total HTTP header and BODY data statistics.



Leaf Server Monitor C-ICAP Summary (expanded view)

5. MYSQL Summary – provides detailed Leaf Intercept module statistics including pages processed per second, peak statistics, average data rates, dropped pages, page write errors, module configuration options, MySQL configuration options, current free space available to the MySQL database, and MySQL configuration options such as target database name and target table name for a particular Leaf module or appliance image.



Leaf Server Monitor MYSQL Summary (expanded view)

Licensing/Anti-Piracy

To prevent piracy and unauthorized copying of the ICAP Leaf Intercept module and other proprietary components, the Leaf module uses a hardware based licensing scheme. The license module checks for specific on-board network adapter chipsets and PCI bus configuration information and uses it to generate a unique key for each target system.

Creating a license file involves two steps. Each time the Leaf Module loads or reconfigures, it regenerates an encrypted seedfile which contains a hardware map of the current system (/etc/c-icap/leaf.seed). This file then can be used to create a license file which is installed on the target system(/etc/c-icap/leaf.license).

Leaf uses an internal use only proprietary command line program or a internal PHP program to create the license file. The license file can be sent or received as an email attachment, or can be remotely downloaded from an internet web server running a Leaf License Server (leaf-www).

The key for decrypting the license file is computed by the Leaf Module from the current systems networking hardware map, then used to decrypt the license file. The license file contains a UUID license code and digital signature which must match for that particular system. If the module is copied to a different system or installed on a new system, the Leaf Intercept Module will require a new license file (/etc/c-icap/leaf.license) in order to initialize and become active. After the /etc/c-icap/leaf.license file has been installed, the C-ICAP server can be reinitialized which will trigger reloading of the Leaf module with the new license file.

The Leaf License Server (leaf-www) is a PHP based web server that manages digital seed and license files organized into customer accounts and ledger history with online payments, and is integrated with paypal and other online payment systems. This server and the licensing tools are internal use only

to run the business customer accounts and provides online access to software license purchases.

Deployment Plan

The intellectual property rights, ownership, and the sub-licensed trademark of the Leaf Linux technology are for sale at a very reasonable price. I am also available to become an employee or contractor for anyone who acquires ownership rights for a reasonable salary and I will continue to refine and enhance the technology.

The technology has been under development since the fall of 2018, and it is now ready for customer deployment in a variety of environments. I have run the technology from local appliances and in cloud based systems. I am presently looking for customers to beta test the technology and harden it. I also need customer input on the web console features to determine what additional functionality may be required to properly support a customers specific environment.

This approach of using commodity database and web technologies has many attractive features which help solve customer problems of enforcing use policies on corporate, campus, or private networks while allowing customers extraordinary flexibility in deployment options and configurations.

Sagecreek Partners has my consent to broker any potential opportunities, including setting pricing, sales, customer placement, beta customer and site selection, or sale of the technology, and of course should receive significant compensation for its assistance if it brings a deal to the table. I have not as of yet determined exactly how much customers should be charged. There are also open issues on whether Leaf should be restricted as only available on a Leaf Appliances or whether the C-ICAP server module can be sold as a software only product for customers with pre-existing Squid Servers.

Given the current architecture, all of these options are possible. Since Leaf can be deployed as a standalone appliance up to and including a fully distributed multi-appliance setup, sale of appliance chassis alone could be quite lucrative to support back-end databases and ICAP server appliances. This is particularly true if Leaf is deployed in a large organization with large numbers web clients which would require large amounts of database storage for daily web traffic monitoring.

Trademark Sublicense

Jeffrey Merkey <jeffmerkey@gmail.com>

Linux Mark Sublicense Application from [Jeffrey Vernon Merkey]

Linux Foundation Programs <trademarks@linuxfoundation.org> Fri, Dec 11, 2015 at 8:02 PM
Reply-To: trademarks@linuxfoundation.org
To: jeffmerkey@gmail.com

Success! Your application has been submitted to the Linux Mark Team.

Thank you, **Jeffrey Vernon Merkey** for your interest in the Linux Foundation Linux Mark Program.

Your request has been sent to our team for review.

We will respond to you as soon as possible.

----- Receipt for Your Records -----

License Number: 20151212-0559
Full Name: Jeffrey Vernon Merkey
Email: jeffmerkey@gmail.com
Phone: 3852992437
Company Name: Individual
Website:
Street Address: 182 N Sun Arbor Terrace Apt 1063
Street Address:
City: Salt Lake City
State: UT
Zip / Postal Code: 84116
Country: United States of America
Previous Sublicense?: 0
Previous Sublicense No:
Sublicense Mark: Leaf Linux
Sublicense Mar k URL:
Goods: Distribution of software that is Linux-based. Distribution of Network Forensics file systems, applications, and services
[Quoted text hidden]

Jeffrey Merkey <jeffmerkey@gmail.com>

Linux Mark Sublicense Request Granted for [20151212-0559]

Linux Foundation Programs <trademarks@linuxfoundation.org> Mon, Jan 25, 2016 at 10:03 AM
Reply-To: trademarks@linuxfoundation.org
To: jeffmerkey@gmail.com

Notification of Granted License: Sublicense has been **granted**

Thank you, **Jeffrey Vernon Merkey** for your interest in the Linux Foundation Linux Mark Program.

Your sublicense request has been reviewed by our team.

License No. **20151212-0559** has been granted!

If you have additional questions, or comments, please feel free to contact us at lmi_query@linuxfoundation.org.

Regards,
The Linux Foundation