



Defcon 2013

Mach-O Malware Analysis:
Combatting Mac OSX/iOS Malware with Data Visualization

Remy Baumgarten

ANRC LLC.

Mach-O Malware Analysis:

Combatting Mac OSX/iOS Malware with Data Visualization

Remy Baumgarten, Security Engineer, ANRC LLC.

Draft Date: Apr 02, 2013

Apple has successfully pushed both its mobile and desktop platforms into our homes, schools and work environments. With such a dominant push of its products into our everyday lives it comes as no surprise that both of Apple's operating systems, OSX and iOS should fall under attack by malware developers and network intruders. Numerous organizations and Enterprises who have implemented BYOD (bring your own device) company policies have seemingly neglected the security effort involved in protecting the network infrastructure from these potential insider threats. The complexity of analyzing Mach-O (Mach object file format) binaries and the rising prevalence of Mac-specific malware has created a real need for a new type of tool to assist in the analytic efforts required to rapidly identify malicious content. In this paper we will introduce *Mach-O Viz*, a Mach-O Interactive Data Visualization tool that lends itself to the role of aiding security engineers in quickly and efficiently identifying potentially malicious Mach-O files on the network, desktop and mobile devices of connected users.

Mach-O Malware Analysis: Combatting Mac OSX/iOS Malware with Data Visualization

Introduction.....	4
Why a New Tool?	5
Introducing Mach-O Viz	6
Demonstrating Mach-O Viz's Features: Analysis of CustomInstaller a.k.a. Yontoo Trojan	11
Analysis of keychain_dumper: iOS Hacker Utility	19
Analysis of MacDefender: OSX's First Malware Threat	25
Mach-O Viz: Where To Go From Here.....	29
Conclusions	29
About ANRC	30
Contact	30

Introduction

These days it is safe to say that Mac is almost everywhere. From our MacBook laptops to our mobile devices such as iPhones and iPads, the prevalence of Mac computing cannot be ignored and makes for a lucrative target for malware authors and potential network intruders. Over the past few years we've seen examples of Mac malware make its way through the prying eyes of the quality assurance engineers at Apple's App Store and subsequently right onto our mobile devices as witnessed in the Flashback Trojan attack last year.¹ To think that these attacks are isolated incidents would be dangerously naive.

Scarier still, even today there is a complete lack of governance in organizations for policing mobile devices connected to their private network infrastructure. In one eye-opening statistic, 57%² of IT Managers had seen mobile related traffic on their networks with no means of actively enforcing their network defense guidelines.

Many IT groups believe it's Apple's responsibility to act as the first line of defense against Mac targeted threats. Though OSX and iOS operating systems have numerous security mechanisms in place to significantly reduce Mac's attack footprint, the assumption that Apple software can prevent all malicious software from appearing on your desktop or mobile devices is far from true.

Making matters worse there is the Jailbreak community who play a cat-and-mouse game with Apple security engineers to circumvent all iOS security mechanisms and allow installation of arbitrary unsigned binaries on mobile devices. How many users in an average organization are bringing in Jail-broken devices to work and connecting them to the local LAN? More importantly, how many of those devices are carrying Malware?

The need for a better understanding of the Mach-O file format coupled with a focus on network security are the primary motivations which led to the development of Mach-O Viz. Mach-O Viz is a data visualization tool specifically created for IT security engineers and malware analysts to tackle the Mach-O file format complexity and provide a rapid analysis solution for Mac software. Mach-O Viz is provided free, multiplatform and employs an array of open source software.

¹ http://www.mercurynews.com/business/ci_22741463/apple-are-mac-computers-becoming-more-vulnerable-malware-virus

² <http://www.forbes.com/sites/markfidelman/2012/05/02/the-latest-infographics-mobile-business-statistics-for-2012/>

Why a New Tool?

At ANRC, we've been tackling the Mach-O malware problem internally in our research and development efforts, so we understand the problems posed by introducing these binaries into networks ill-equipped to process them through their standard network defense systems. With most of the world practicing a Windows-centric ideology with regard to security appliances, coupled with a ridiculously complex file specification from Apple, we felt the need to delve deeper into the problem by:

- Examining and evaluating the existing tools available to help decipher the Mach-O format.
- Finding working examples of security products equipped to process Mach-O malware.
- Attempting to find a tool that could analyze these files regardless of the underlying architecture.
- Researching a better way to view the file internals of Mach-O files.

We examined any Mach-O tool we could find to help aid the analysis of potentially malicious binaries that could be on i386, x86_64 or ARM architectures. The focus on these targets was intentional to make sure the binaries were likely generated using XCode, Apple's development IDE, or llvm/gcc. The following chart lists the results of our initial research efforts and helped drive our design and development for Mach-O Viz.

Tool	Graphic	Multiple Architectures	Network Security Aware	Easy to Understand	Ease of Use
IDA Pro	Yes (sometimes)	Yes	No	No	No
otool	No	Yes	No	No	No
class-dump	No	Objc Only	No	Yes	Yes
Machoview	Yes	Yes	No	No	Yes
ptool	No	Yes (old/no support)	No	No	Yes
otool-ng	No	Yes (old/no support)	No	No	No
hopper	Yes (sometimes)	Yes	No	No	No

Figure 1. Evaluation of existing Mach-O tools (Green=Meets Need, Red=Does Not Meet, Yellow=Some Need Met)

The design for Macho-Viz centered on fusing the advantages offered by these tools and then adding a focus on network security as well. Ultimately the goal is to help the network defender understand the Mach-O file format better and provide a method to effectively and efficiently analyze a particular binary for malicious behavior. One of the essential elements in the Macho-Viz tool design was the need for a simple interface supporting a powerful and accurate data analytics engine that would yield results comparable to an advanced

disassembler. Additionally we wanted to make the tool free to the public to help increase security awareness and benefit the reversing community that is struggling to keep up with malicious Mach-O binaries.

Introducing Mach-O Viz

Mach-O Viz was developed to introduce the idea of presenting a Mach-O binary **visually** which in turn makes it easier for anyone to see how the file is constructed, i.e. how it is formed from the header through the load commands and into all of its corresponding segments. In addition to visualizing the file itself we wanted a powerful back-end graph visualization and analytics system for graphing the binary's disassembly for the top 3 platforms: i386, x86_64 and ARM6/7. The fundamental component was to keep this process as visual as possible while maintaining a simple-to-use interface.

With data visualization in mind, the next challenge was to make the tool as cross-platform as possible. The desire was for client access to the interface to be simple regardless of platform (mobile, desktop, android, osx, iOS ...etc). With mobility and cross-platform interoperability important requirements, we decided to implement Mach-O Viz as an HTML5/JavaScript front end using a Mac OSX Server backend. Additional design features include:

- Use any client capable of running HTML5/JavaScript, thereby significantly increasing the types of devices that could make use of Mach-O Viz.
- Keep the back-end as "Mac" as possible. We wanted to rely on Apple's updates of the Mach-O spec and its tools, such as otool, in their native environment. This would keep Mach-O Viz updated and relevant by default.
- Gain access to the LLVM disassembler for the most accurate ASM we can feed into our analytics engine.
- Make use of as many Open Source utilities that added benefit as possible.

The initial interface designed morphed several times until we found a common ground that included both the File Structure Visualization and the Function Graphing Visualization. The combination of these produced a simple-to-navigate interface where visual interactivity was key (Figure 2).

The File Structure Visualization was developed in a drill-down type of style capable of driving the user deeper into the fields represented by the major segments. For example, clicking "Load Commands" in the 1st level tier would drill down so that you see a visual representation of all of the individual load commands and could subsequently drill even deeper into an individual command (Figure 3). The File Structure Visualization interfaces with the Function Graphing Panel, as well, allowing you to dump various segments into its Hex Editor for easy viewing. In addition, the files __TEXT segment is automatically analyzed and graphed by a powerful analytics-graphing engine (Figure 4).

Mach-O Malware Analysis: Combatting Mac OSX/iOS Malware with Data Visualization

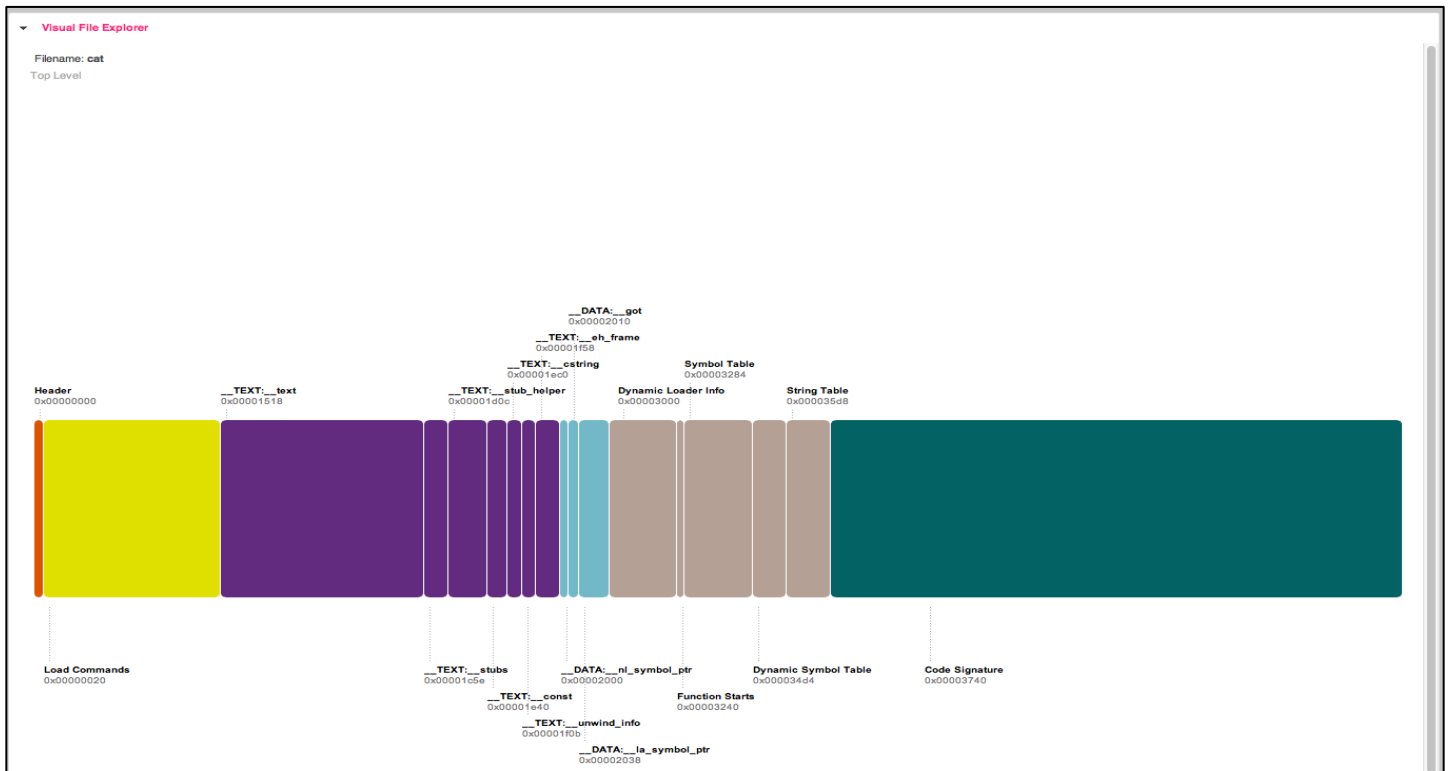


Figure 2. File Structure Visualization.

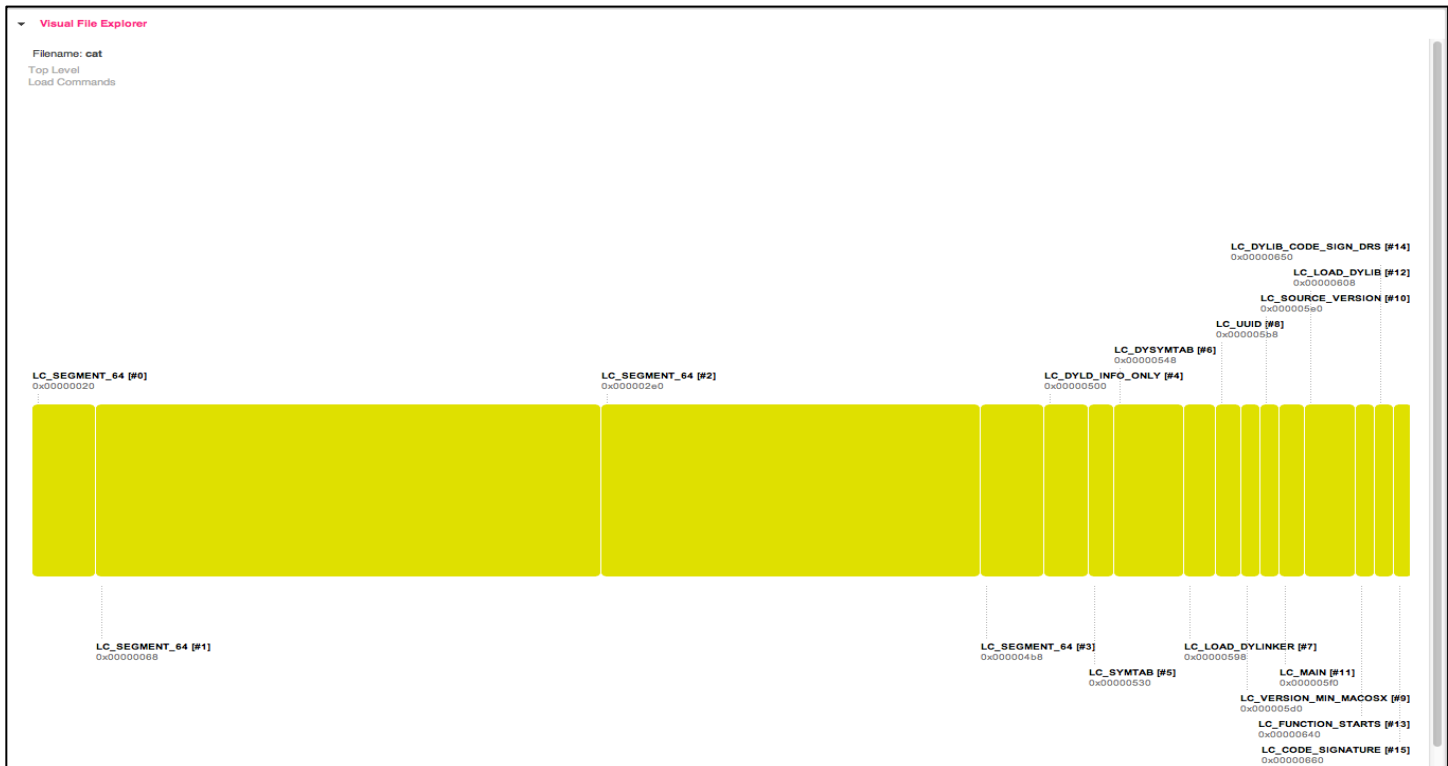


Figure 3. Drilling Down into Load Commands

Mach-O Malware Analysis: Combatting Mac OSX/iOS Malware with Data Visualization

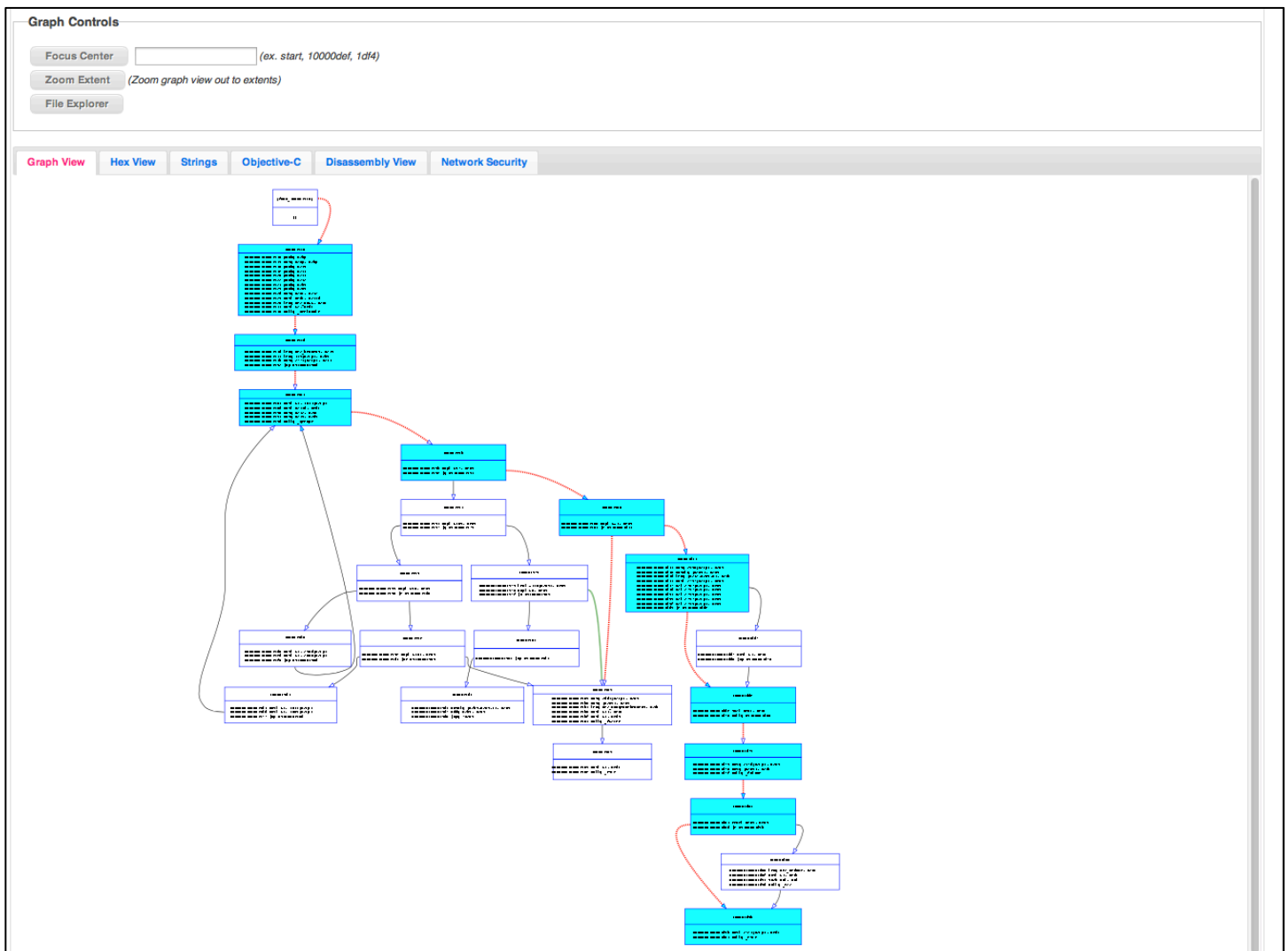


Figure 4. Interactive Function Graphing Visualization

The Function Graphing Visualization provides an interactive method of navigating the disassembly of the binary's __TEXT segment while focusing on functional code blocks, i.e. sequences of basic blocks that perform a specific task. In addition to the interactive graph component, we also introduced methods for processing and analyzing the data for display in context specific Views: Hex View, Strings, Objective-C, Disassembly View and Network Security.

The data and static information segments can be displayed in the Hex View while Strings provides a breakdown of code-referenced data. For example, to see all the strings available in the binary, a user can click between the segments "CString" and "String Table" and display those in Hex View. Clicking the "Strings" tab displays only those strings that have been resolved in the code itself and ultimately show up in the Graph Visualization.

Mach-O Malware Analysis: Combatting Mac OSX/iOS Malware with Data Visualization

To view Objective-C data structures, the Objective-C tab provides this information using the open source tool Class-dump³, which was been integrated into Mach-O Viz's interface.

The Disassembly View provides a paginated and colored interface to the file's disassembly. This allows a user to see all of the instructions (not just the function graphs) in a clickable fashion.

Mach-O Viz, with a heavy emphasis on the information security risk of Mach-O binaries, focuses on Security by introducing 2 unique features:

1. Identifying code segments which are using API's and Functions flagged as **Security Risks**.
2. Identifying and automatically generating network and static file signatures for the binary.
 - Mach-O Viz does this in 2 ways:
 - a) By detecting network domains, ip addresses, urls & web protocols embedded in the binary.
 - b) Calculating a unique binary signature for the file itself using Mach-O MAGIC value in the file's header plus a unique 16 bytes from the binary's String Table.

The images below (Figure 5, 6) demonstrates these features:

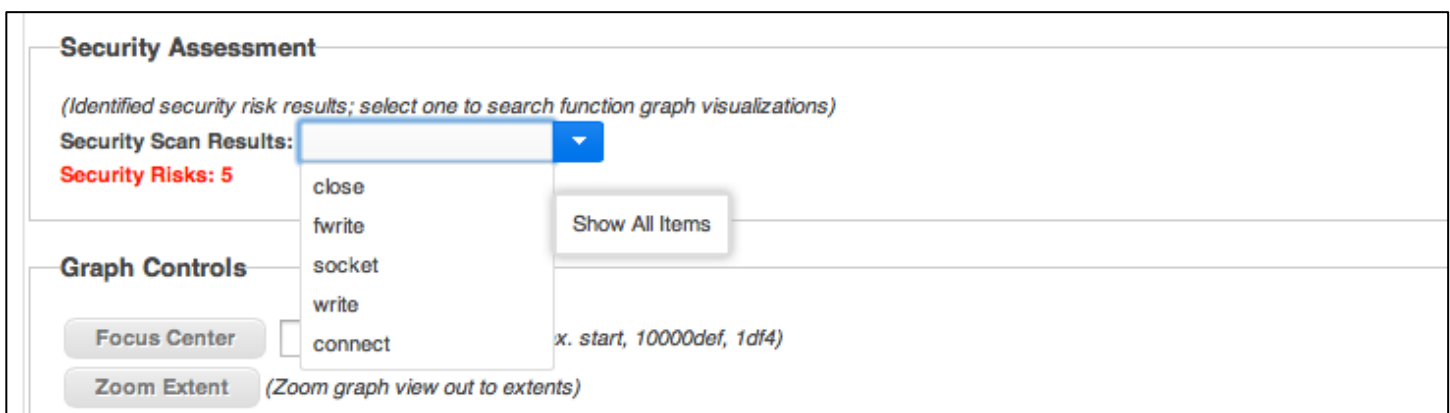


Figure 5. Flagging risky API's and functions in use by the Mach-O binary.



Figure 6. Automated signature generation for arbitrary Mach-O files.

Mach-O Viz separates itself from more complicated tools, such as debuggers and disassemblers, by displaying the file's structure as graphically as possible while still providing the underlying assembly structure for more advanced users (Figure 7).

³ <http://stevenygard.com/projects/class-dump/>

Mach-O Malware Analysis: Combatting Mac OSX/iOS Malware with Data Visualization

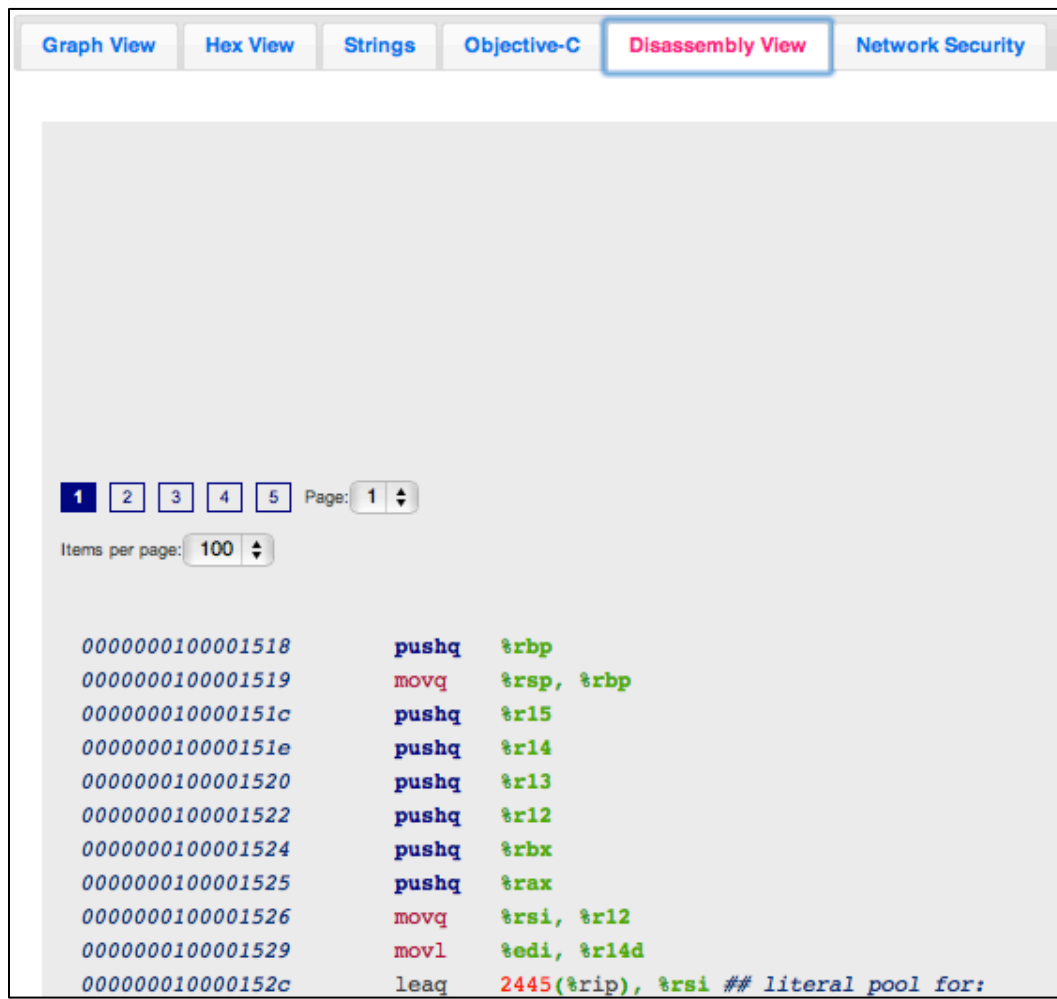


Figure 7. Disassembly View available for in-depth analysis.

Demonstrating Mach-O Viz's Features: Analysis of CustomInstaller a.k.a. Yontoo Trojan

To demonstrate the effectiveness of Mach-O Viz against current malware and also exercise some of its features, we'll analyze a relatively new sample called the Yontoo Trojan. This nasty piece of malware "installs itself as a browser plug-in and infects Google Chrome, Firefox, and Safari Browsers via Adware."⁴

This malware sample also demonstrates that Mac operating systems, OSX in this case, are still very much vulnerable to infection. The Yontoo Trojan relies on the user to enable the infection by masquerading as a browser HD video plugin (Figure 8) to social engineer the user into downloading and installing it. This tactic is not new. Fooling an unsuspecting user into opening malicious attachments and downloads to compromise their own systems continues to be a technique used by attackers because it's simple and it works.

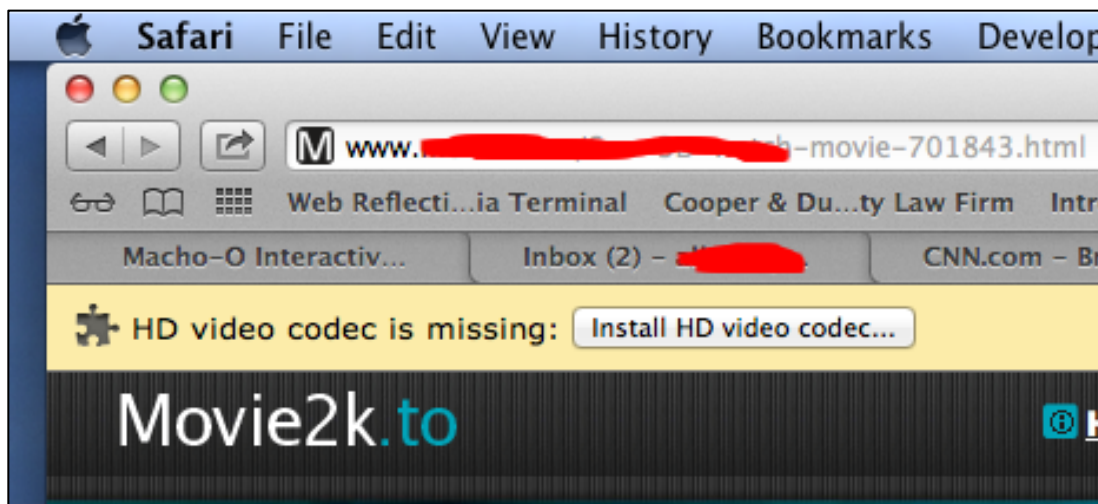


Figure 8. CustomInstaller social engineering the user into installing a "HD" codec.

Uploading the file to Mach-O Viz will kick start the automatic analysis, which lasts between several seconds to several minutes depending on the size of the file and its complexity (ex. number of functions). During this process Mach-O Viz performs the following functions:

- Parses the file structure to build the file visualization's header, load commands, text and data segments.
- Scans and detects basic blocks, and from those basic blocks builds functions.
- Builds DOT (GraphViz format) files for the graph visualization.
- Resolves strings, names and ObjC components in the code (__TEXT) segment.
- Conducts a security scan of the file and, where applicable, builds automated SNORT signatures for static file and network traffic detection.

⁴ <http://www.ibtimes.com/yontoo-trojan-new-mac-os-x-malware-infects-google-chrome-firefox-safari-browsers-adware-1142969>

Mach-O Malware Analysis: Combatting Mac OSX/iOS Malware with Data Visualization

Once the initial analysis concludes, we are presented with Mach-O Viz's interactive user interface showing both file and the function graph visualizations (Figure 9). In addition, the Security Assessment section warns us of 15 identified risks (based on method, function or API) that we should focus our attention on (Figure 10).

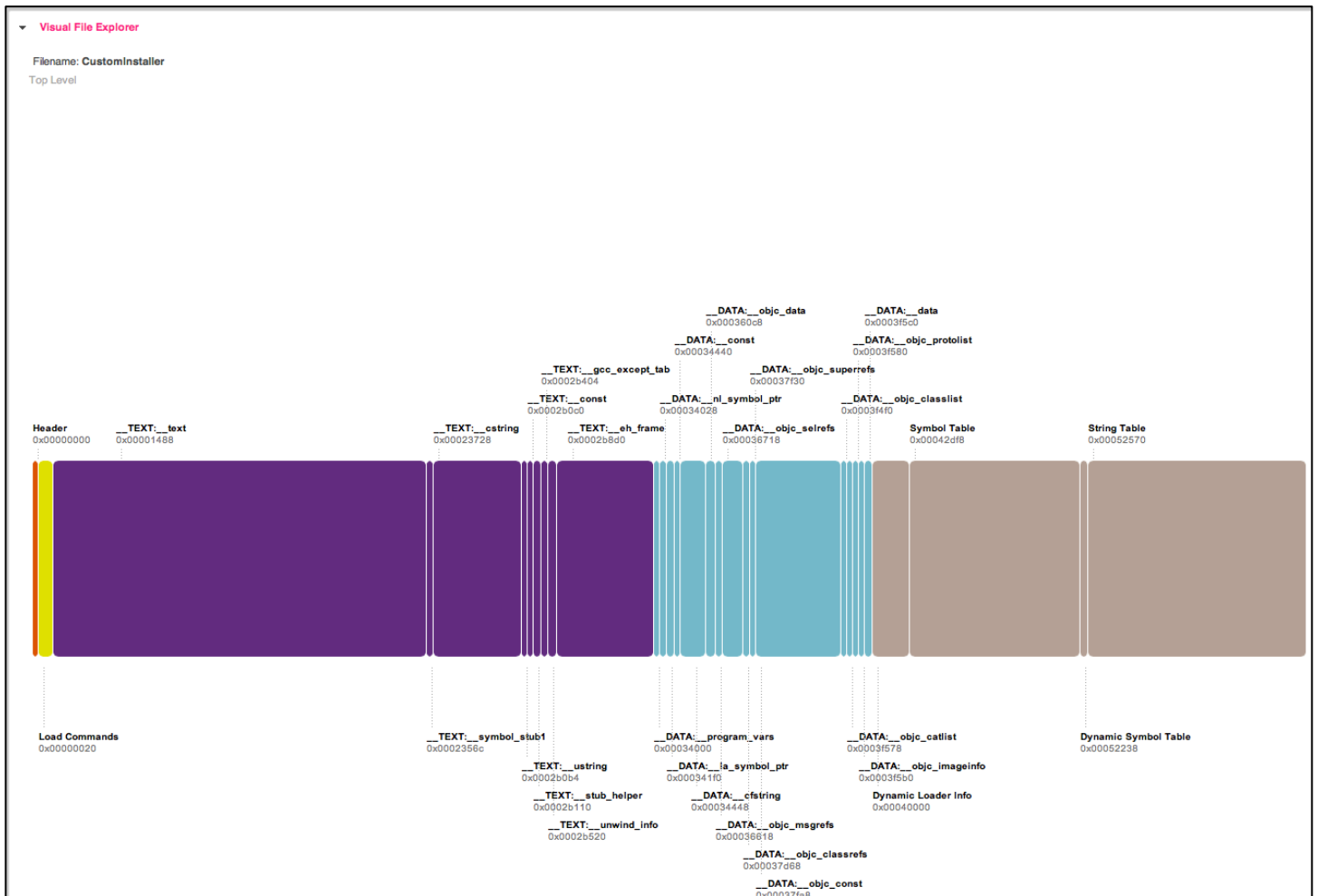


Figure 9.

The function graph visualization automatically calculates the program's entry point and places us at the "main" or start of the *CustomInstaller* program (Figure 11).

Mach-O Malware Analysis: Combatting Mac OSX/iOS Malware with Data Visualization

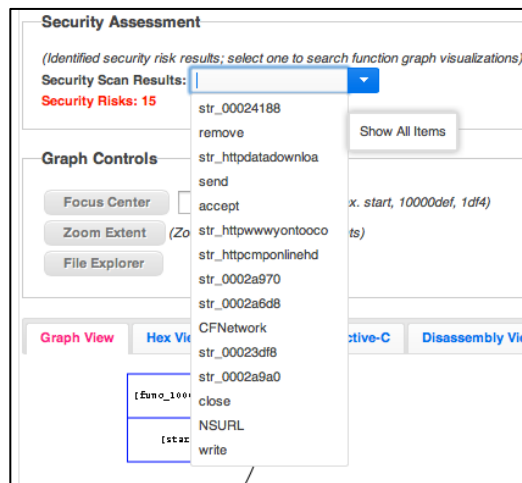


Figure 10. Security Assessment identifies 15 risks that should be examined in further detail.

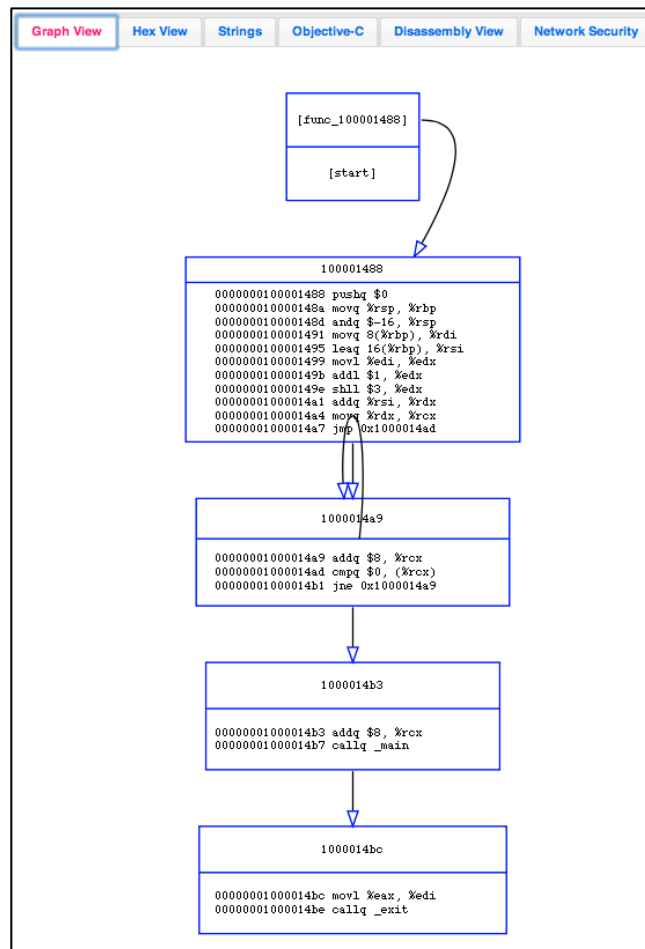


Figure 11. The function visualization calculates and displays the start/main in graph view.

Continuing to view the analysis results, we find under the Strings tab four code-referenced string values which allows us to further highlight the trojan's potential malicious nature (Figure 12).

Mach-O Malware Analysis: Combatting Mac OSX/iOS Malware with Data Visualization

Graph View	Hex View	Strings	Objective-C	Disassembly View	Network Security
str_host			host		
str_html			html		
str_http			http		
str_httpcmponlinehd			http://cmp.online-hd.tv/fitmac.zip		
str_httpdata downloa			http://data.downloadstarter.net/country.asp?st=0		
str_httpwwwyontoo			http://www.yontoo.com/PrivacyPolicy.aspx		
str_https			https		
str_id			id		
str_IfModifiedSince			If-Modified-Since		
str_IfNoneMatch			If-None-Match		
str_InstalledExtens			Installed Extensions		

Figure 12. Mach-O Viz's analysis engine locates important strings and resolves them to the code segment.

The Objective-C view provides us information about harvested data structures, including their implementation addresses. This provides us with a powerful combination to zero in on major functions of interest to analyze. In addition, these data structures hint at the true nature of this binary. For example, we find an interface declaration for an “ExtensionsInstaller” that clearly references installation support for Firefox, Chrome, and Safari. Class-dump (Figure 13) also provides us with their implementation address that can be placed into the graphing visualization for display.

```
Graph View  Hex View  Strings  Objective-C  Disassembly View  Network Security

@end

@interface ExtensionsInstaller : NSObject
{
}

+ (id)extensionValue:(id)arg1:(id)arg2:(id)arg3;          // IMP=0x0000000100022103
+ (id)getFirefoxActiveProfileDir;                        // IMP=0x00000001000221ed
+ (BOOL)installFirefoxExtension:(id)arg1;                // IMP=0x0000000100022490
+ (BOOL)installChromeExtension:(id)arg1;                 // IMP=0x0000000100022815
+ (BOOL)installSafariExtension:(id)arg1;                 // IMP=0x00000001000229db
+ (void)uninstallExtensions:(id)arg1;                    // IMP=0x0000000100022f9d

@end
```

Figure 13. Objective-C tab provides important information on declarations and data structures.

Taking this valuable information, we can search this address, for example, *0x1000229db*, which is the method that appears to install the Yontoo trojan browser extension into Safari, into Mach-O Viz's Interactive Function Search and generate the function graph.

Mach-O Malware Analysis: Combatting Mac OSX/iOS Malware with Data Visualization

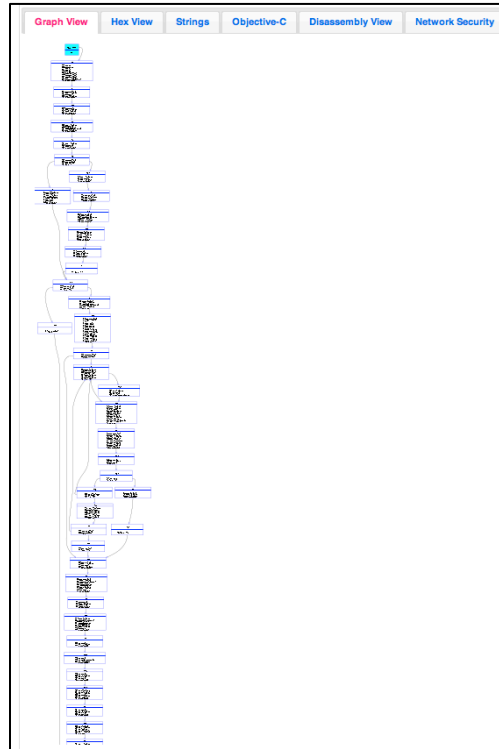


Figure 14. Zoomed out view of function 0x1000229db, implementation function for Yontoo installation into Safari.

We can further dissect this function to find out how the browser extension is installed by leveraging the in-depth analysis and x86_64 string resolution already conducted by Mach-O Viz. The function begins by locating the path to the Safari Extensions that have been resolved correctly (Figures 15,16).

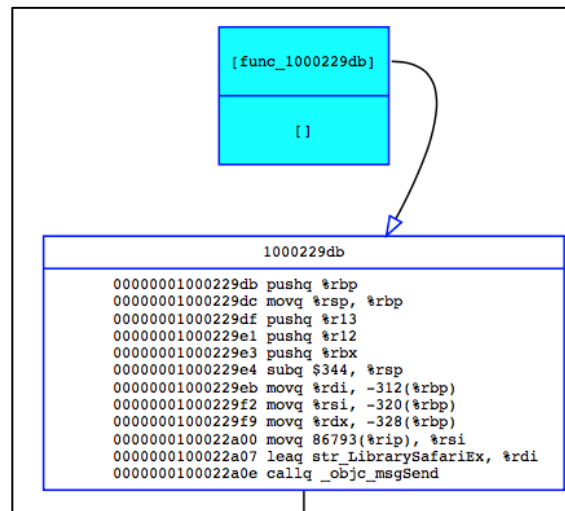


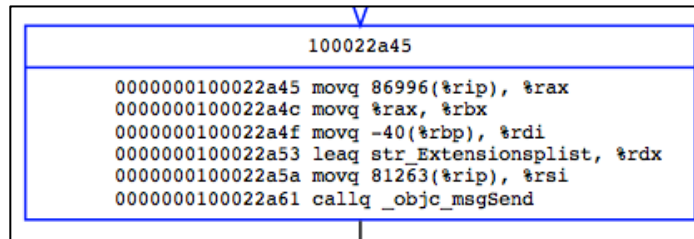
Figure 15. String address for Safari extension path correctly resolved during initial analysis.

str_LibrarySafariEx	-/Library/Safari/Extensions/
---------------------	------------------------------

Figure 16. Auto-generated string “str_LibrarySafariEx” identifies user’s Safari extension directory.

Mach-O Malware Analysis: Combatting Mac OSX/iOS Malware with Data Visualization

Once the Extension directory is located, the installer proceeds to update Safari's Extensions.plist (Figure 17,18) and then copies and enables the new extension to successfully complete the install.

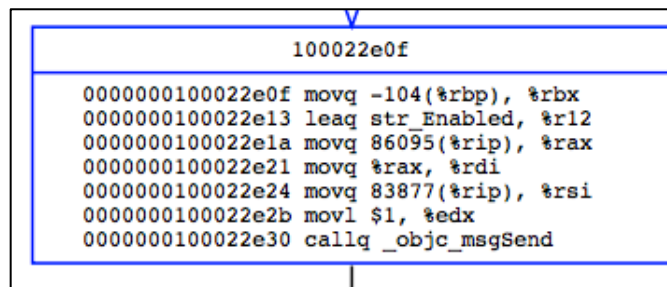


```
100022a45
00000000100022a45 movq 86996(%rip), %rax
00000000100022a4c movq %rax, %rbx
00000000100022a4f movq -40(%rbp), %rdi
00000000100022a53 leaq str_Extensionsplist, %rdx
00000000100022a5a movq 81263(%rip), %rsi
00000000100022a61 callq _objc_msgSend
```

Figure 17. Auto-generated string “str_LibrarySafariEx” identifies user’s Safari extension directory.

str_Extensionsplist	Extensions.plist
---------------------	------------------

Figure 18. Strings tab shows us the string in question.



```
100022e0f
00000000100022e0f movq -104(%rbp), %rbx
00000000100022e13 leaq str_Enabled, %r12
00000000100022e1a movq 86095(%rip), %rax
00000000100022e21 movq %rax, %rdi
00000000100022e24 movq 83877(%rip), %rsi
00000000100022e2b movl $1, %edx
00000000100022e30 callq _objc_msgSend
```

Figure 18. Auto-generated string “str_Enabled” enables the new extension in Safari.

Finally, we can see the edge case wherein if the plugin already exists the installation routine for Safari jumps past the code to install and enable the plugin and simply exits the function. Mach-O Viz’s ability to dynamically zoom in and out easily draws this branch to our attention and allows us to “color” the code path in question (Figure 19).

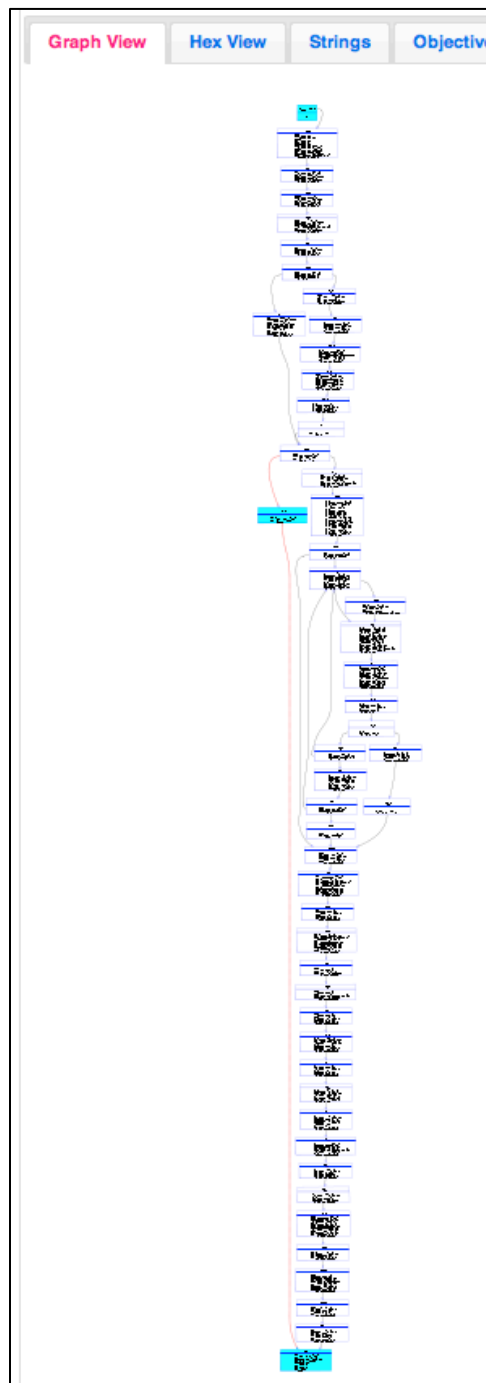


Figure 19. Code highlighting the branch that finds the Yontoo plugin installed and exits.

To find cross-references to this Safari extension installation function, we can simply search under “Names/XRefs” in the Interactive Graph Function Search panel for the address `0x1000229db`. Alternatively, we could have made our way to this function by searching under “Strings” for occurrences of the string “str_Extensionsplist”.

Mach-O Malware Analysis: Combatting Mac OSX/iOS Malware with Data Visualization

Interactive Graph Function Search

Functions: Names/XRefs: Strings: Search Results:

of Identified Functions: 620
Lines of Assembly: 32517

Figure 20. Exercising some of Mach-O Viz's interactive search capabilities to locate the same function.

The Network Security tab (Figure 21) is equally important. In this case Mach-O Viz has provided us with a wealth of SNORT style signatures clearly targeting both the network traffic generated by the Yontoo Trojan, as well as a static file signature to prevent future infection. For the network security administrator, simply uploading the Yontoo Trojan binary *CustomInstaller* into Mach-O Viz and navigating to this Network Security tab would provide immediate feedback and assistance in detecting and containing this real and very recent Mac OSX malware threat rapidly until a more thorough analysis and signature could be created (if necessary).

Graph View Hex View Strings Objective-C Disassembly View **Network Security**

Automated Network and Static File Signatures for "CustomInstaller"

```
str_0002a9a0: alert tcp any - (msg:"Mach-O Viz Automated Scan: Web Protocol Embedded"; flow:established; offset: 0; content:"http://www.yontoo.com"; classtype:policy-violation; sid:11111111; rev:1;)
str_00024188: alert tcp any - (msg:"Mach-O Viz Automated Scan: Web Protocol Embedded"; flow:established; offset: 0; content:"http://data.downloadstarter.net/"; classtype:policy-violation; sid:11111111; rev:1;)
str_0002a970: alert tcp any - (msg:"Mach-O Viz Automated Scan: Web Protocol Embedded"; flow:established; offset: 0; content:"http://www.yontoo.com"; classtype:policy-violation; sid:11111111; rev:1;)
str_0002a6d8: alert tcp any - (msg:"Mach-O Viz Automated Scan: Web Protocol Embedded"; flow:established; offset: 0; content:"http://cmp.online-hd.tv/"; classtype:policy-violation; sid:11111111; rev:1;)
str_00023df8: alert tcp any - (msg:"Mach-O Viz Automated Scan: Web Protocol Embedded"; flow:established; offset: 0; content:"http://data.downloadstarter.net/"; classtype:policy-violation; sid:11111111; rev:1;)

File Signature: MAGIC+String Table: alert tcp any - (msg:"Mach-O Viz Automated Scan: File Mach-O MAGIC:String Table 16 bytes"; flow:established; file_data; content:"ICF FA ED FE"; depth:4; content:"120 00 2D 5B 41 70 70 44 65 6C 65 67 61 74 65 20"; depth:337280; classtype:policy-violation; sid:11111111; rev:1;)
```

Figure 20. Mach-O Viz's Network Security tab auto-generates valuable network defense signatures for us.

Analysis of keychain_dumper: iOS Hacker Utility

Keychain Dumper⁵ is a popular utility for dumping all passwords saved within iOS's keychain. Installation of this utility assumes the iOS device (iPhone, iPad, etc.) has been jailbroken. Analyzing this file with Mach-O Viz will exercise the disassembly analysis engine's ability to handle ARM7 instructions while providing Objective-C and String information, when applicable.

After uploading the keychain dumper binary into Mach-O Viz and drilling down into the header, we can verify that it has been compiled as an ARM architecture (Figure 21).

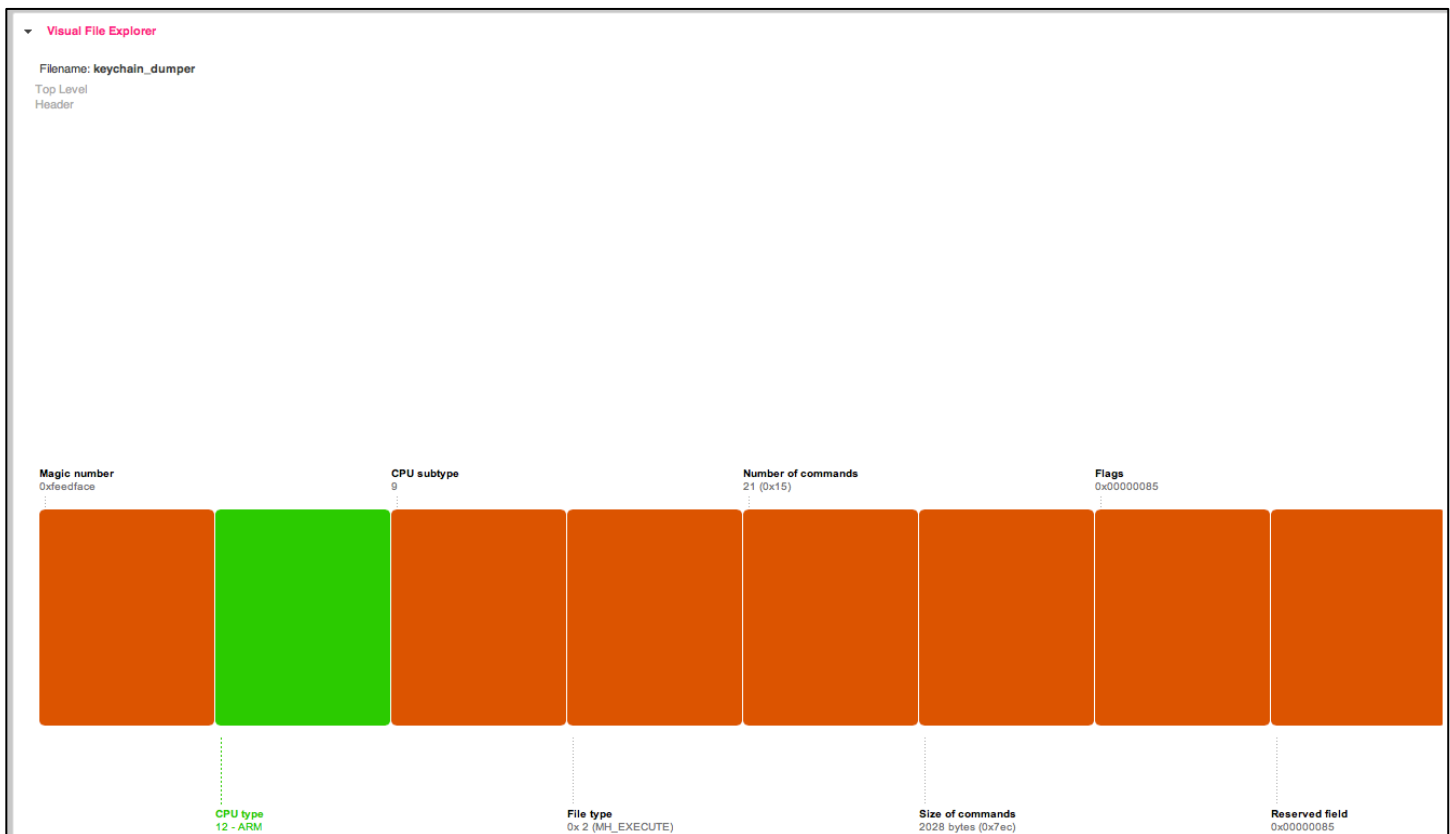


Figure 21. Keychain dumper header shows ARM architecture as the target for this binary.

In addition, we can verify that the file has not been code signed by examining the ENCRYPTION_INFO load command (Figure 22). Also notice as we drill into the load commands the “bread crumbs” in the top-left give us a simple history of how we managed to navigate to this field. This useful feature allows simple navigation of the Mach-O file format load commands, which can become cryptic and complicated in a standard console data dump. By presenting the same information in a visual file structure, we can easily find the fields of interest and concentrate only on those.

⁵ <https://github.com/ptoommey3/Keychain-Dumper>

Mach-O Malware Analysis: Combatting Mac OSX/iOS Malware with Data Visualization

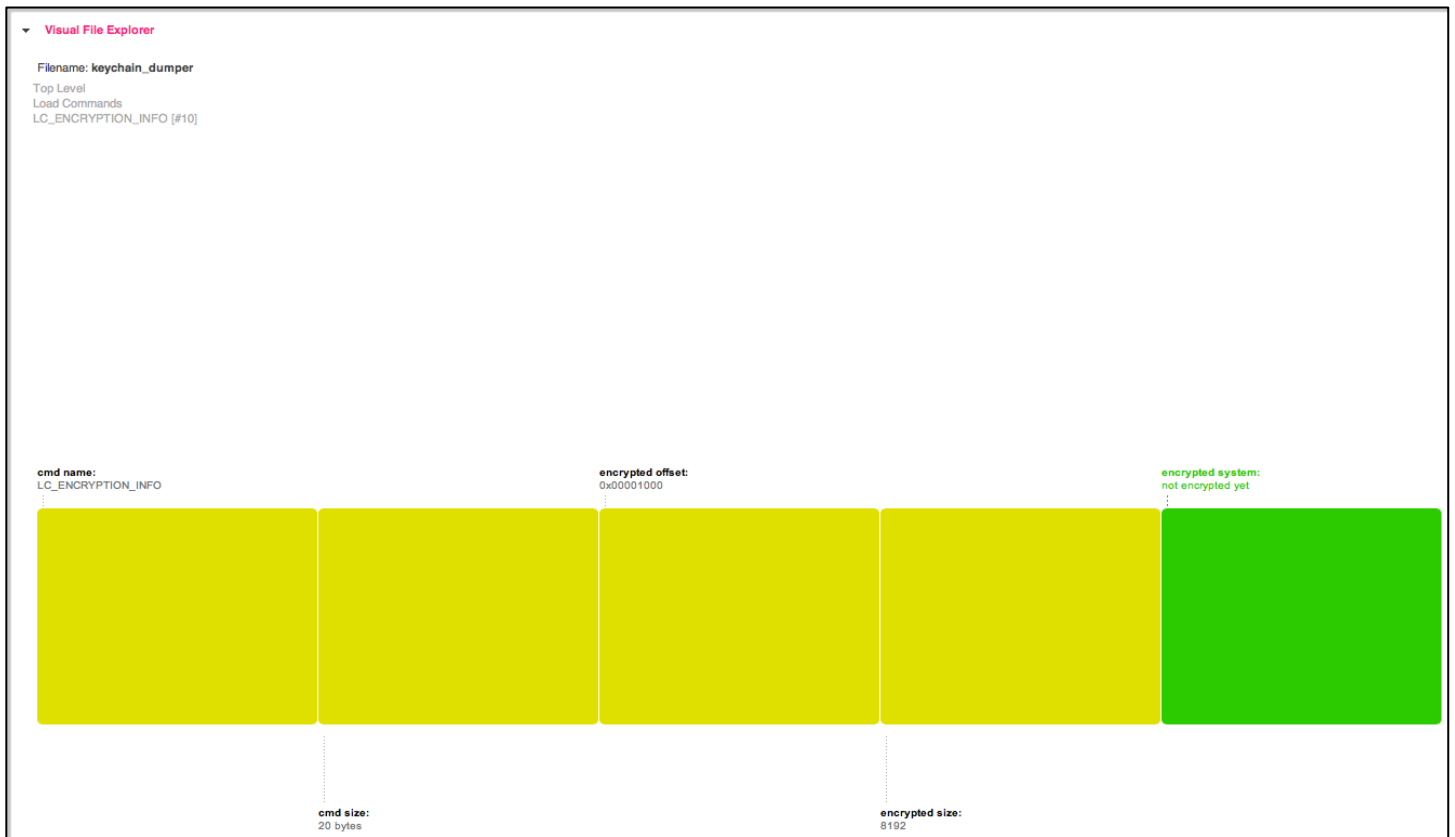


Figure 22. Drilling down into the load commands we arrive at ENCRYPTION_INFO and validate an unencrypted ARM binary.

Continuing the analysis of this hacker utility, our Security Risk score is 4, which is quite low. From a network security threat perspective you definitely do not want this utility floating around; however, compared to a malware Trojan or similar backdoor utility, this binary doesn't appear to display malicious API usage. Our sole string flagged as a security risk turns out to point to Apple's domain.

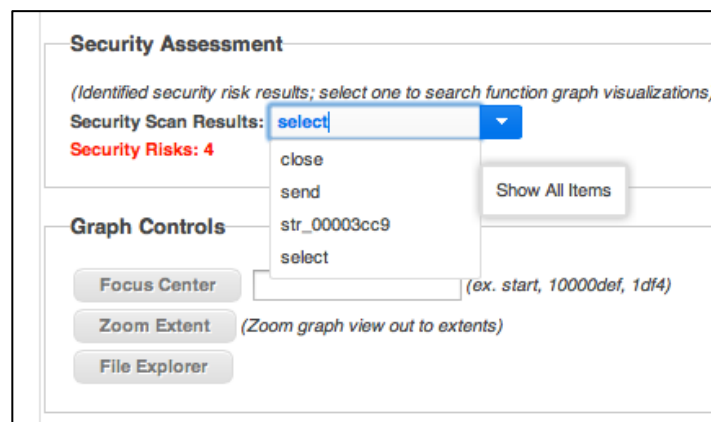


Figure 23. A low amount of security risks demonstrates that lack of this binary's ability to act as a true threat.

Mach-O Malware Analysis: Combatting Mac OSX/iOS Malware with Data Visualization

A powerful capability of Mach-O Viz is to enumerate and resolve all Objective-C string data for method names and variables. This allows a complete enumeration of most of the code and turns malicious code analysis into technical reading. Case in point, the “Names/XRefs” (Figure 24) of the keychain dumper utility systematically gives a general idea of its inner workings and true capability.

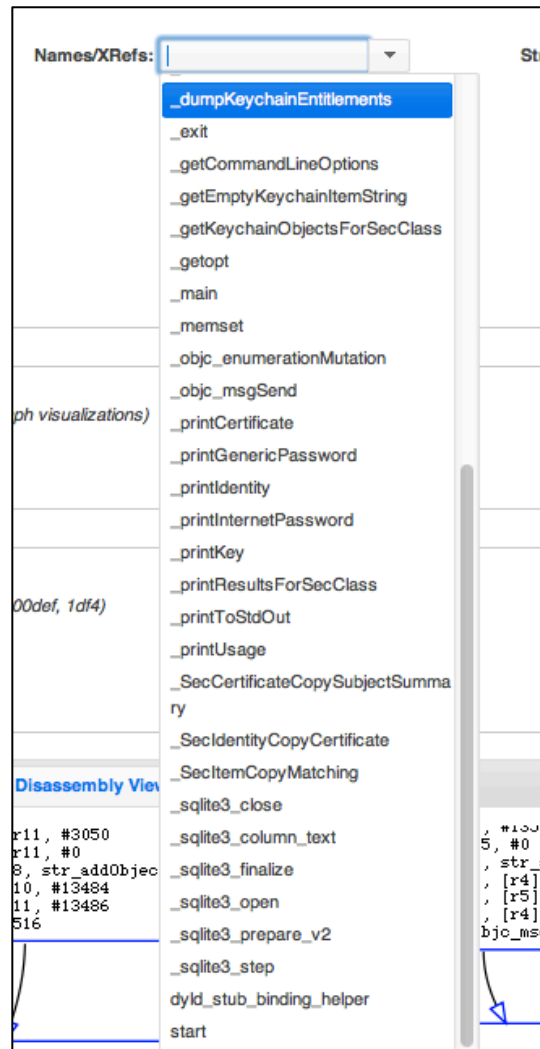


Figure 24. Objective-C structure and method enumeration provides immediate value in quickly triaging this binary.

The following methods highlight the core of this binary: *dumpKeychainEntitlements*, *printCertificate*, *printGenericPassword*, *printIdentity*, *printInternetPassword* and *printKey*. The *sqlite* methods provide data access to the keychain database store. In terms of functionality you can examine these functions to confirm they actually perform “as advertised”.

Mach-O Malware Analysis: Combatting Mac OSX/iOS Malware with Data Visualization

Let's conduct a verification of the `dumpKeychainEntitlements` method to confirm it does what it says. Selecting it from the "Names/XRefs" and then from "Search Results" brings the method into the Graph Visualization tab (Figure 25).

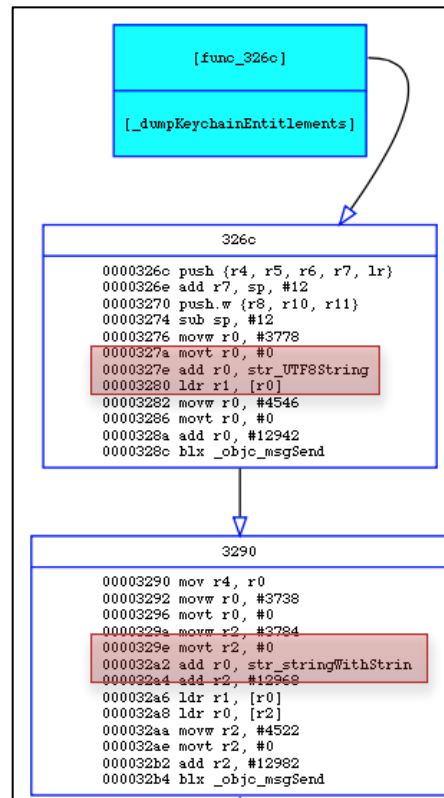


Figure 25. A string resolution algorithm was developed to resolve ARM7 string references in Mach-O Viz.

It's important to note that in order to perform ARM string resolution in the code, an instruction-tracing algorithm was developed in order to resolve these references for Mach-O Viz's graphs. A native "*otool*" code dump will not provide this amazingly useful information to you. The figure below (Figure 26) shows an Apple *otool* dump of the same code sequence without the string references.

Mach-O Malware Analysis: Combatting Mac OSX/iOS Malware with Data Visualization

```
705 _dumpKeychainEntitlements:
706 0000326c b5f0 push {r4, r5, r6, r7, lr}
707 0000326e af03 add r7, sp, #12
708 00003270 e92d0d00 push.w {r8, r10, r11}
709 00003274 b083 sub sp, #12
710 00003276 f64060c2 movw r0, #3778
711 0000327a f2c00000 movt r0, #0
712 0000327e 4478 add r0, pc
713 00003280 6801 ldr r1, [r0]
714 00003282 f24110c2 movw r0, #4546
715 00003286 f2c00000 movt r0, #0
716 0000328a 4478 add r0, pc
717 0000328c f000ea7e blx 0x378c @ symbol stub for: _objc_msgSend
718 00003290 4604 mov r4, r0
719 00003292 f640609a movw r0, #3738
720 00003296 f2c00000 movt r0, #0
721 0000329a f64062c8 movw r2, #3784
722 0000329e f2c00200 movt r2, #0
723 000032a2 4478 add r0, pc
724 000032a4 447a add r2, pc
725 000032a6 6801 ldr r1, [r0]
726 000032a8 6810 ldr r0, [r2]
727 000032aa f24112aa movw r2, #4522
728 000032ae f2c00200 movt r2, #0
729 000032b2 447a add r2, pc
730 000032b4 f000ea6a blx 0x378c @ symbol stub for: _objc_msgSend
731 000032b8 a902 add r1, sp, #8
732 000032ba 4605 mov r5, r0
733 000032bc 4620 mov r0, r4
734 000032be f000ea7e blx 0x378c @ symbol stub for: _objc_msgSend
```

Figure 26. Apple's otool doesn't provide the deep code analysis of Mach-O Viz.

Further down the method we find the call to open the keychain database along with a string referencing a SELECT statement (Figure 27). Our Strings tab quickly reveals the details of this call (Figure 28).

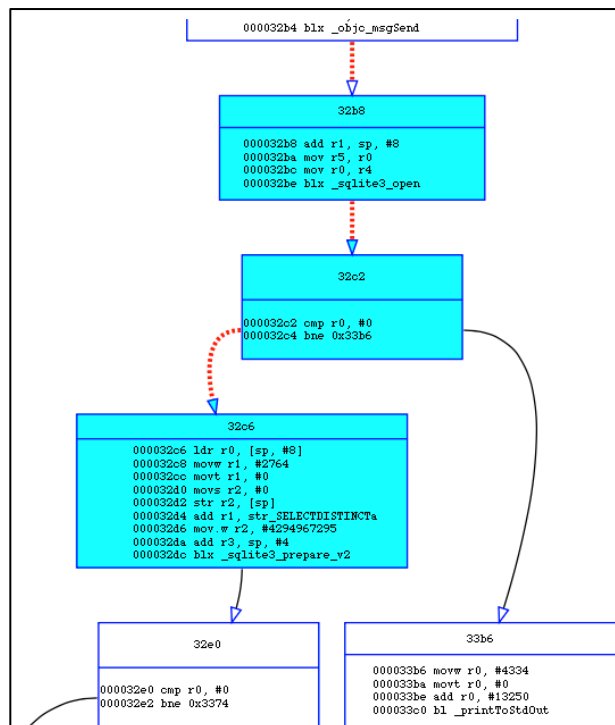


Figure 27. Tracing the code path of the `dumpKeychainEntitlements` method.

Mach-O Malware Analysis: Combatting Mac OSX/iOS Malware with Data Visualization

str_SELECTDISTINCTa	SELECT DISTINCT agrp FROM genp UNION SELECT DISTINCT agrp FROM inet
---------------------	---

Figure 28. Only due to Mach-O Viz's string resolution ability were we able to easily track down this value.

The next several code blocks iterate through the results of the SELECT statement and build a string out of the data returned. The graph visualization allows us to color and observe this code loop (Figure 29).

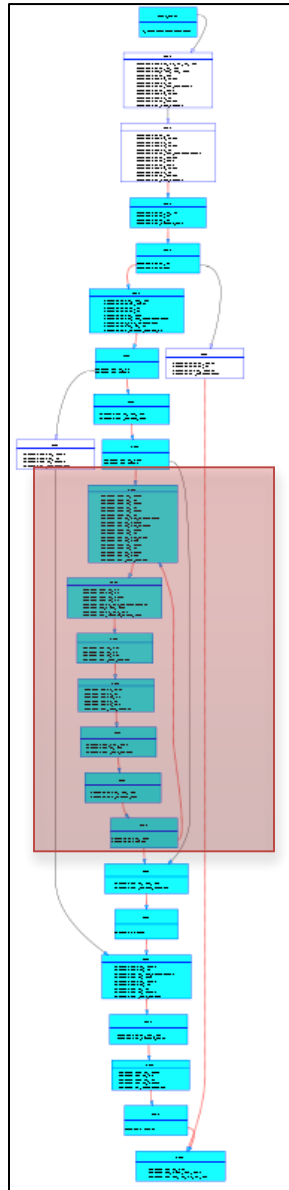


Figure 29. Graph view shows us the SELECT statement and subsequent code loop to aggregate the results as strings.

The resulting string data is dumped to STDOUT and the function exits. As we have demonstrated in a matter of a few minutes, you can quickly triage this file as a hacker utility, grab and deploy its flat file signature (Network Security tab) and move on to other more malicious binaries.

Analysis of MacDefender: OSX's First Malware Threat

MacDefender quickly solidified itself as the first real threat to the OSX operating system back in 2011. Operating under the principles of social engineering, an unsuspecting user is lured into installing it as a legitimate Mac Anti-Virus product. It then attempts to get the user's credit card number by asking them to pay for the "full" version. It also hijacks the user's browser to display sites related to pornography.⁶

MacDefender is a good example of a FAT file structure whereby a binary is compiled for multiple architectures and executes on the one detected by the Mach-O loader. Sending the file into Mach-O Viz illustrates the two architectures supported, x86_64 and i386 (Figure 30). Visually we can also see that the x86_64 binary is larger than its i386 counterpart while the header is barely a sliver when weighed against the actual files.

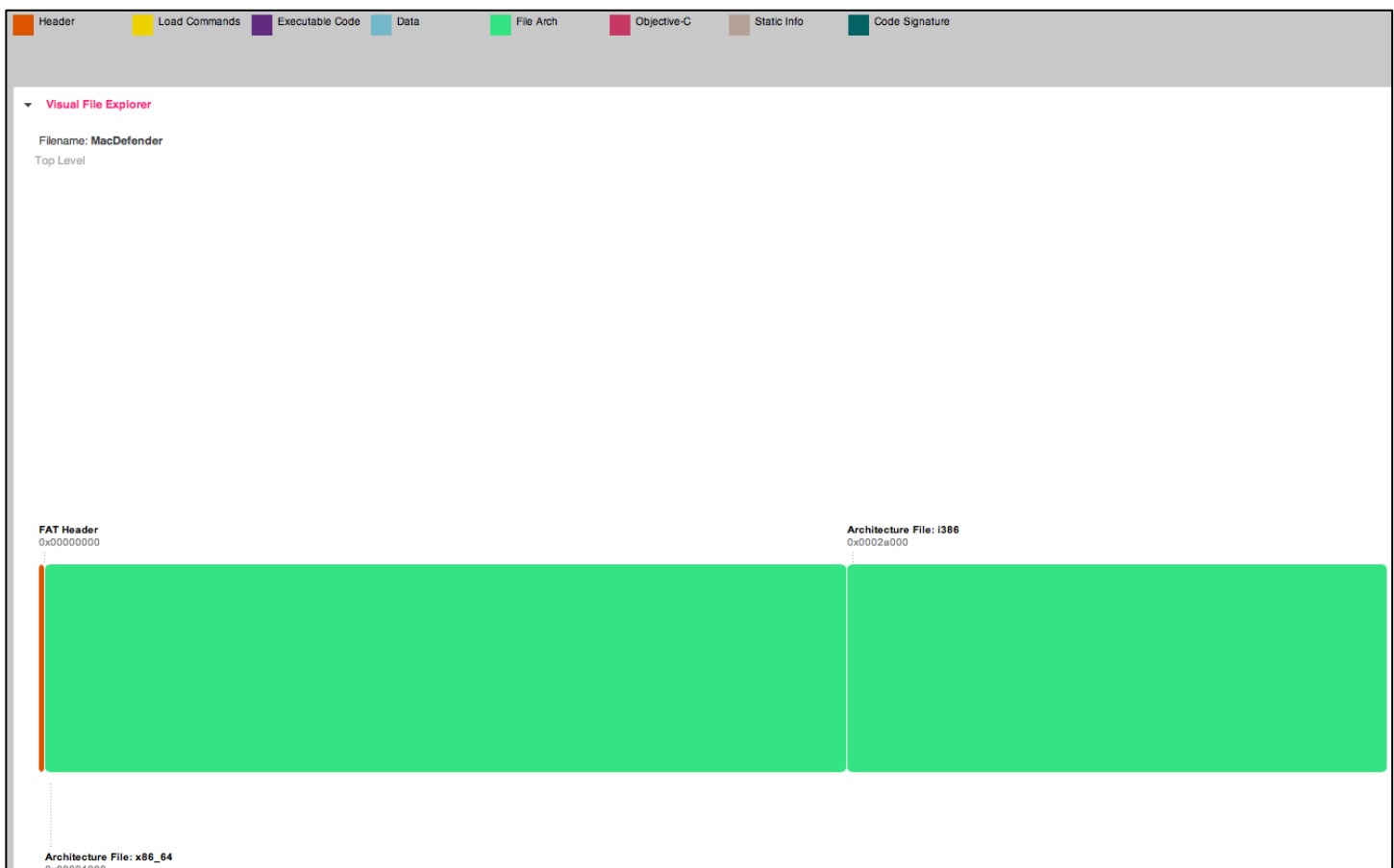


Figure 30. Mach-O Viz correctly parses and display FAT file headers with multiple architectures.

The Security Risk field and Network Security tab provide us with immediate feedback as to the true nature of this binary by highlighting the malicious IP addresses and domains embedded within (Figure 31).

⁶ http://en.wikipedia.org/wiki/Mac_Defender

Mach-O Malware Analysis: Combatting Mac OSX/iOS Malware with Data Visualization

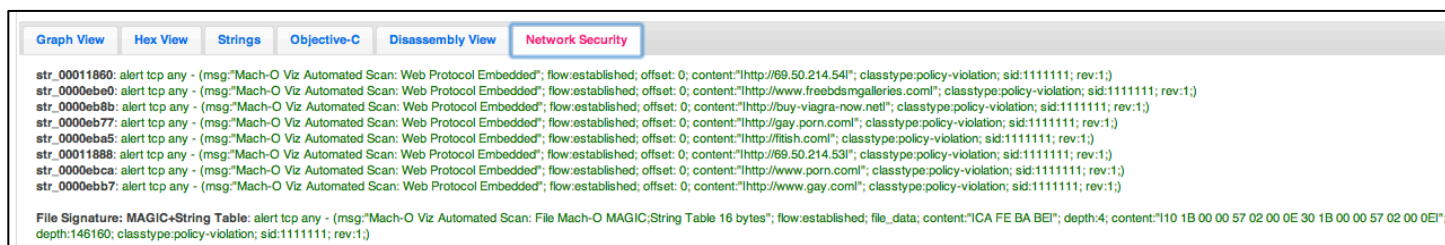


Figure 31. Malicious URL's and IP addresses embedded within MacDefender provide quick insight into its real intention.

The `sysctl` API provides access to get/set kernel level attributes and rightfully scores as a significant security risk (Figure 32) in the automated security assessment.

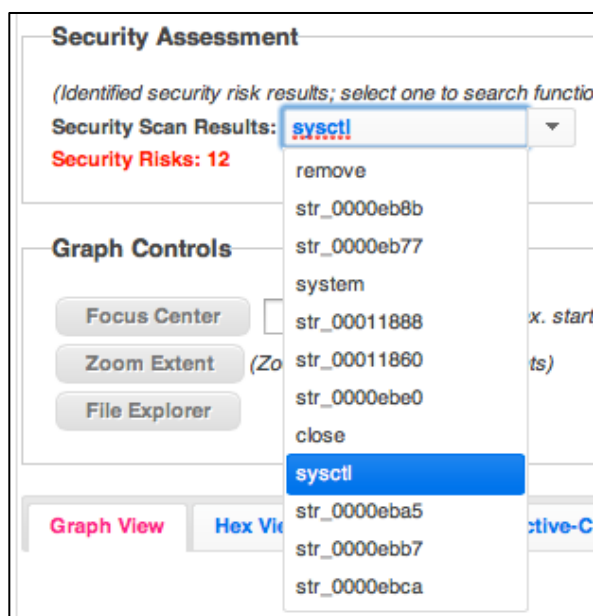


Figure 32. Sysctl is not an API function you want your average OSX binary to be accessing.

Ironically, the “fake” anti-virus comes equipped with the bells and whistles to make you think that it is in fact a legitimate product as seen in the “Names/XRefs” list (Figure 33).

Mach-O Malware Analysis: Combatting Mac OSX/iOS Malware with Data Visualization

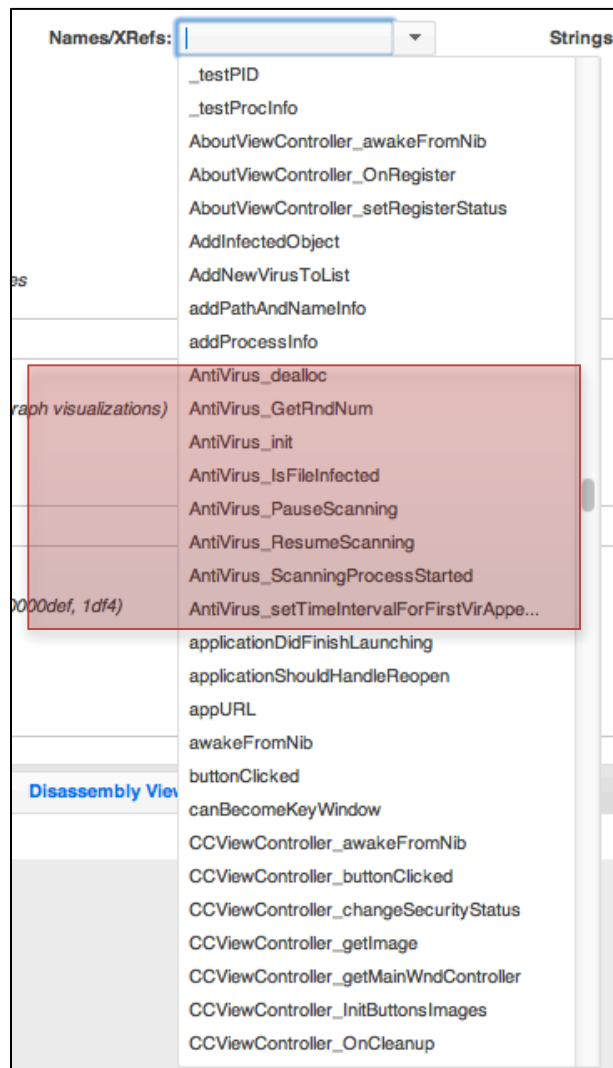


Figure 33. This malware author created routines to mimic an actual Anti-Virus scan.

As part of the scam to fool the user, we can examine one of the AV's functions, *AntiVirus_IsFileInfected*. In a normal anti-virus this would be a complicated feat consisting of signature based detection and heuristics to detect maliciousness of a particular binary. The unusually small size of this AV's detection function points us to something else however (Figure 34).

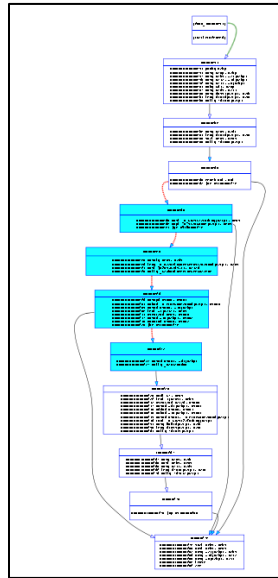


Figure 34. The world's smallest AV file infection detection routine.

Focusing in on the highlighted code blocks we see the use of a random number generator to create the effect of a delayed scanning (Figure 35) in order to make it appear as if it is actually finding viruses while it displays fake names to the user. Examining more of the so-called functionality reveals more of same scamming.

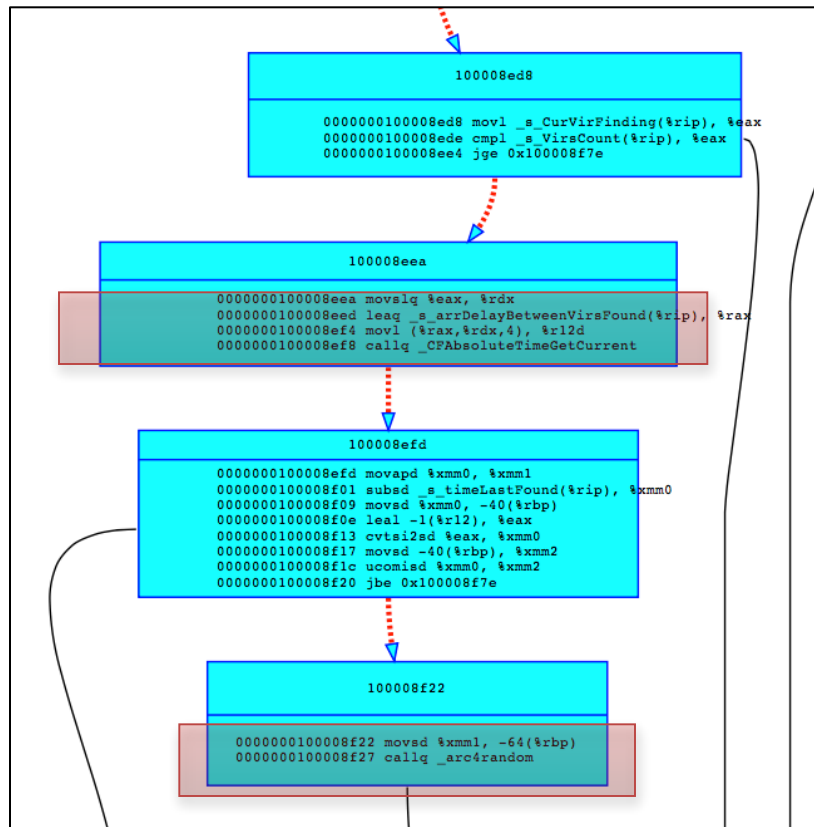


Figure 35. Fake AV using random delays to create the appearance of a scan.

Mach-O Viz: Where To Go From Here

As we've demonstrated, Mach-O Viz can clearly be used as an effective tool for analyzing Mac related malware both for the OSX and iOS operating system using data visualization. In addition to the many features already implemented, we'd like to include the following functionality in future versions:

- Archiving support for previously analyzed files.
- Take Mach-O Viz into the Cloud for inline automated scanning of Mach-O files for Enterprise networks.
- Function charting across multiple binaries looking for matching code sequences.
- Visually mapping Mach-O Viz's file and graph structures into an active debugger such as LLVM.
- Plugin support for modular enhancements.

Conclusions

Mach-O Viz was developed to fill the need to properly and easily conduct malware analysis on Mac related malware regardless of the architecture or device. By creating a terminal like interface using HTML/JavaScript and developing a powerful back-end analytic engine we've managed to build a unique and extremely useful tool to quickly triage Mach-O binaries regardless of their format, visually display them and provide unique and automated signatures for deployment to network defense systems.

The ability for network security staff and analysts to react quickly and accurately to the latest Mac threats is critical, especially in today's Mac-centric world. Mach-O Viz provides the capability required to easily make this happen without sacrificing the power of a full-featured commercial disassembler.

About ANRC

ANRC delivers advanced cyber security training, consulting, and development services that provide our customers with peace of mind in a fast-paced and complex cyber security environment.

ANRC was formed with two visions in mind: to provide the best and most current computer security education possible, and to administer that education through a revolutionary new approach, endowing our customers with knowledge that will truly be usable, valuable, and retainable.

Contact

Mr. Remy Baumgarten
Security Engineer, ANRC LLC.
1-800-742-7931