

17/03/2011



TRUSTDEFENDER  
LABS

## APPLE MAC OS X AND MALWARE – MYTH VS REALITY



[WWW.TRUSTDEFENDER.COM](http://WWW.TRUSTDEFENDER.COM)

© Copyright 2011 by Symbiotic Technologies Pty Ltd. All rights reserved. No part of this publication may be reproduced or stored in a retrieval system or transmitted in any form or by any means, without the prior written permission of the copyright holder. Symbiotic Technologies Pty Ltd is the sole copyright owner of this publication.

TrustDefender Labs Report | Nick Blievers / Andreas Baumhof

# APPLE MACOSX AND MALWARE – MYTH VS REALITY

## 1 Table of Contents

1	Table of Contents .....	2
1	Executive Summary .....	3
1.1	HellRaiser.....	4
1.2	HellRaiser Protocol.....	6
1.3	HellRaiser Limitations.....	7
2	Apple’s Answer: File Quarantine .....	7
3	XProtect.....	8
3.1	XProtect in detail.....	9
4	The good .....	11
5	The bad .....	11
5.1	What programs have LSFileQuarantineEnabled set?.....	11
5.2	Some attack vectors .....	12
5.3	Performance considerations.....	12
6	Conclusion.....	13

# APPLE MACOSX AND MALWARE – MYTH VS REALITY

## 1 Executive Summary

There has been much debate on whether Apple's Mac OS X operating system is more secure by design or have taken advantage of security through obscurity, as Macs have traditionally had much lower penetration into the market than Windows based systems.

However, as Apple has enjoyed rapidly growing success over the past few years, not only with its iPhones and iPads but with also the growing popularity of Macs and Mac OS X, it comes as no surprise to see security companies predict that attacks on Apple's systems and its users will also start increasing dramatically.

While 'Security' seems to be a competitive advantage, Apple has realised that some dedicated malware defences are needed, so it shipped an 'Anti-Malware' component with its Mac OS X updates. In June 2010, 3 (three) malware programs were on the list<sup>1</sup>. When we checked in March 2011, there were still three!

Recently, Sophos made waves announcing a new "backdoor Trojan" for the Mac OS X platform, (<http://nakedsecurity.sophos.com/2011/02/26/mac-os-x-backdoor-trojan-now-in-beta/>). Although this Trojan (called Black Hole) is very new, it is not in a state that poses any threat, and doesn't provide much for us to look at yet in an in-depth report.

Given that, we thought this a perfect opportunity to look at an existing piece of Mac malware known as HellRaiser 4.2, chosen because it is detected by Apple's anti-malware defences, unlike "Black Hole". Using HellRaiser 4.2 lets us see how Apple's defences work, so we can test their effectiveness and discover their limitations.

There is very little public information on the technical details of Mac OS X's anti-malware feature, however, we will provide some technical details here. Furthermore after testing Apple's anti-malware defences, you might have guessed that we found several ways around the built-in protection within just a few hours.

At 'TrustDefender Labs', we typically look at the most sophisticated Trojans available. While ongoing research on the latest malware Trojans has led us to have very high expectations on the levels of sophistication now seen in advanced malware, the Mac OS X HellRaiser 4.2 Trojan left us quite disappointed by being a relatively simple Trojan that, despite some "clever" features, doesn't look like it was written by professionals.

Even though the Mac OS X malware industry is still in its infancy, the whole situation provides cause for concern going forward, especially as many of Apple's end-users aren't accustomed to treating malware as a serious security risk given Apple's legacy as safe technology that's easy to use and master, with historically miniscule and harmless malware threats compared with the barrage of malware, viruses and Trojans that Windows users have been subjected to for years. Introduction – Mac OS X and Malware

---

<sup>1</sup> [http://www.readwriteweb.com/archives/apple\\_quietly\\_updates\\_mac\\_anti-malware\\_feature.php](http://www.readwriteweb.com/archives/apple_quietly_updates_mac_anti-malware_feature.php)

# APPLE MACOSX AND MALWARE – MYTH VS REALITY

Apple and malware are two words that aren't commonly spoken together. However, this is set to change in a big way in the near future. While Mac OS X only holds a relatively small percentage of the desktop market, its close smartphone and tablet cousin iOS is in the hands (literally!) of millions of people. This is a tempting target that will see increasingly sophisticated and successful attempts at exploitation.

For all doubters, Apple has just released iOS 4.3, a massive update to iOS where Apple patched a stunning 49 (!) security vulnerabilities. Apple notes that these vulnerabilities “may lead to an unexpected application termination or arbitrary code execution”, which in plain English means having your device owned. The full list is available at Apple's support site <http://support.apple.com/kb/HT4564>. It's important to note that these vulnerabilities left iOS users unprotected until a new version of iOS was released, as opposed to having updates released on a much more timely cycle.

To give you an idea of the seriousness of unpatched vulnerabilities, we only need to look at one of the most targeted attacks on Windows operating systems and applications – Stuxnet.

The perpetrators behind Stuxnet exploited seven different zero-day attacks, developing highly successful and highly targeted malicious software to break into what would otherwise be highly secure installations. They went to incredible lengths to accomplish what they wanted. Do you think Stuxnet would have been less successful if the Siemens SCADA machines were controlled by Apple Macs?

This is the reality that we have to face in the future: there are inherent risks with what we do, regardless of the operating systems we use.

The trick is how you deal with the risk and whether you can manage the risks. This is the purpose of this TrustDefender Labs report, which looks at Apple's built-in malware defences.

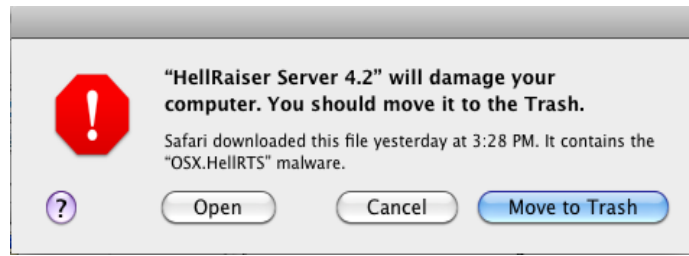
## 1.1 HellRaiser

One of the existing pieces of malware on Mac OS X is HellRaiser 4.2. As we wanted to test Apple's built-in protection, we needed to look at a Trojan that Apple already provides protection against, with HellRaiser 4.2 fitting the bill. We'll look at it in this section a bit in more detail and will then look at Mac OS X's defences.

Hellraiser is actually about 12 months old, with Virus Total reporting just over a 50% hit rate, so this Mac OS X Trojan isn't new or unknown by any stretch of the imagination. However, we figured it's still interesting to investigate, both in terms of what is capable of, and, more interestingly, what Apple is doing to protect against these kinds of things.

First up, let's just see what happens if we run it:

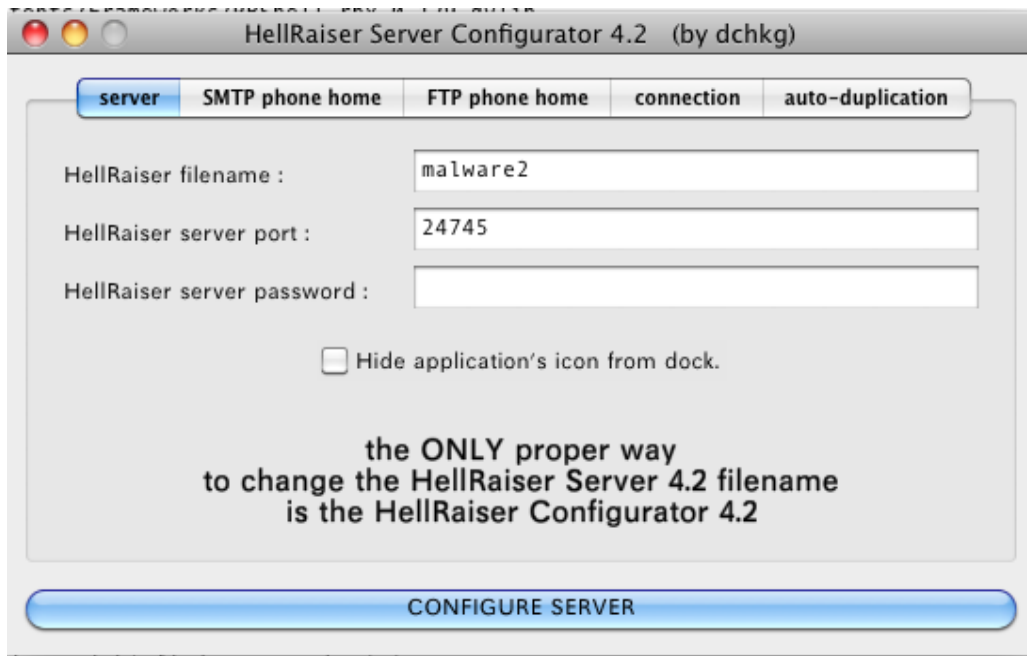
## APPLE MACOSX AND MALWARE – MYTH VS REALITY



Well, that's a good start. It looks like Apple's built in protection detects it!

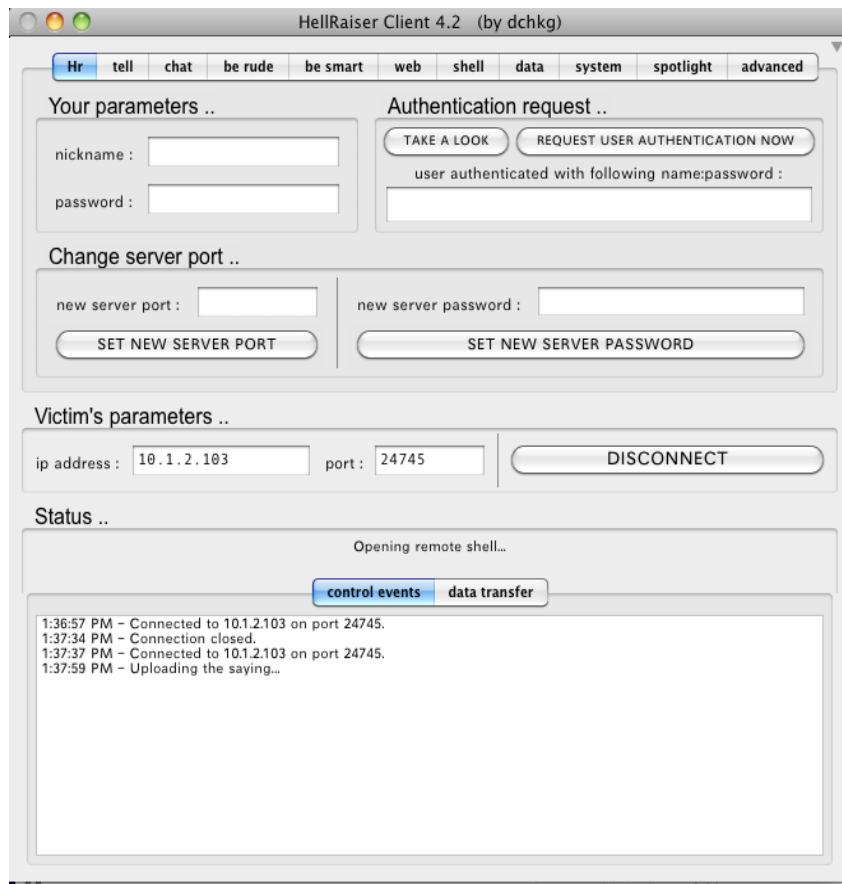
However, let's put that aside for a moment, and have a quick look at this malware. It's not new, and seems pretty rudimentary, but offers a few clever features. HellRaiser is a Remote Administration Tool (RAT). The goal is to get the server running on a target system by whatever means necessary (i.e. social engineering), and then be able to connect to it from the RAT client.

Configuration is straightforward:



Once configured, installed and running on the target system, the server will email or FTP its local IP address and port to the specified host. From here, the cyber criminal can connect using the client software:

# APPLE MACOSX AND MALWARE – MYTH VS REALITY



## 1.2 HellRaiser Protocol

As you can see from the last image in the previous section, the HellRaiser client exposes quite a bit of functionality. The server and client communicate via a protocol that appears to be roughly plain text, albeit turned into hex and base64 encoded. For example, if we use the "tell" function to create a message on the target system, Wireshark reports the following message on the main control port:

```
5655784a56446f7848314e46546b51634d446b784d5455324e69353661584146486a  
4d344e6830354e7a6b3044516f3d
```

We can convert this into something a little more useful:

```
$ perl -le 'print pack("H*",  
"5655784a56446f7848314e46546b51634d446b784d5455324e69353661584146486  
a4d344e6830354e7a6b3044516f3d");' | openssl base64 -d
```

```
ULIT:1SEND0911566.zip3869794
```

# APPLE MACOSX AND MALWARE – MYTH VS REALITY

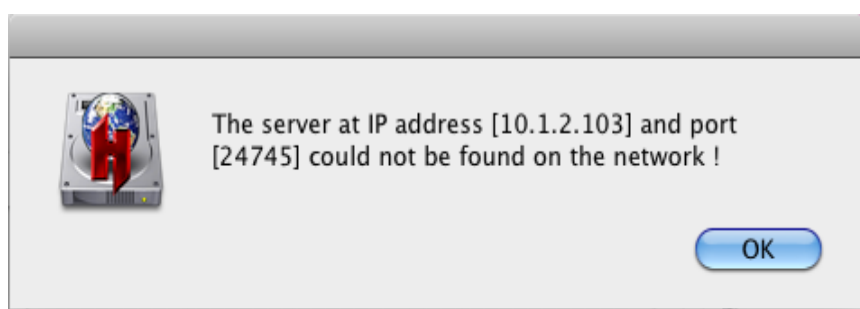
The information circled in red above shows that, over a data port, a zipped file is sent that contains a file named 0911566 which contains our message.

## 1.3 HellRaiser Limitations

For the most part, HellRaiser works but has some quite serious limitations. Firstly, the software is somewhat buggy:

```
3/03/11 1:28:21 PM malware2[2818] malware2(2818,0xa0c21500) malloc: *** error for object 0x112b5c0: double free
*** set a breakpoint in malloc_error_break to debug
3/03/11 1:28:21 PM malware2[2818] malware2(2818,0xa0c21500) malloc: *** error for object 0x112c9e0: double free
*** set a breakpoint in malloc_error_break to debug
```

Secondly, given it waits for a client to attach to the server, any form of NAT or firewall will stop it. Apple's built in firewall stops the connection, as will most routers that perform some kind of Network Address Translation (NAT).



## 2 Apple's Answer: File Quarantine

So, that's what the cybercriminal is attempting. Now let's look at what Apple's OS X security experts are doing to protect us. When a file is downloaded via Safari (or, indeed, any application that has "LSFileQuarantineEnabled" set to true), it is flagged with an extended attribute "com.apple.quarantine". This contains some information about how the file was acquired, specifically the timestamp, application name, and the bundle identifier.

For example:

```
com.apple.quarantine: 0000;4d7468de;Google\x20Chrome;66B1A56C-4296-420D-A895-48DEA91D76F4|com.google.Chrome
```

When a file that has been downloaded is executed, "LaunchServices" will see this flag and prompt the user the first time the application is launched.

# APPLE MACOSX AND MALWARE – MYTH VS REALITY



## 3 XProtect

Only once a file is marked quarantined does Mac OS X's built-in malware protection kick in. There is no official statement as to what this is called, so we and others in the security industry are calling it XProtect, after the name of the definition file. XProtect is configured and updated by a standard plist file, which is a human-readable XML file format, containing a blacklist signature database that all users of Internet security and anti-virus software on Windows systems are well aware of.

Looking in the malware definition list (XProtect.plist), we can see a few entries for HellRaiser:

```
</dict>
<dict>
  <key>Description</key>
  <string>OSX.HellRTS</string>
  <key>Matches</key>
  <array>
    <dict>
      <key>MatchType</key>
      <string>Match</string>
      <key>Identity</key>
      <data>qK+o5ka9agLPqoRHNbLMUIILufU=</data>
      <key>MatchFile</key>
      <dict>
        <key>NSURLTypeIdentifierKey</key>
        <string>com.apple.application-file</string>
      </dict>
    </dict>
  </array>
  <key>LaunchServices</key>
  <dict>
    <key>LSItemContentType</key>
    <string>com.apple.application-file</string>
  </dict>
</dict>
```

This is a really useful protective feature out of the box, and as long as the OS is kept up-to-date, Mac OS X users get some level of protection. On the downside, this list is updated relatively infrequently, at least when compared to anti-virus engines. Some anti-virus vendors claim hourly or even more frequent updates, while Apple's anti-malware definitions are fairly static. Currently only three malware types are part of the list.

Once the amount of malware targeting Mac OS X increases, Apple's infrequent updating may prove to be insufficient, and it's obviously no help against zero-day, click-jacking or session based trojans.

# APPLE MACOSX AND MALWARE – MYTH VS REALITY

## 3.1 XProtect in detail

Let's examine what the XProtect definitions actually mean, and let's see if we can understand the extent of matching that this allows.

XProtect.plist is one small file with a description of the different ways to detect malware. The malware detection is based on the quarantine approach outlined in the last section, which means that **before** any quarantined file is executed, XProtect will check whether to allow that file to run or not. The attributes that XProtect uses to determine whether a file is "bad" will be outlined below, but it is solely based on a blacklist. There is no heuristics or behavioural engine, no zero-day protection, no nothing. Once the application passes the blacklist, there are no further checks beyond the standard quarantine message.

Even if some of the rules defined in XProtect would match, a binary that has no quarantine flag will not have these checks applied.

Straight away, this leads to some attack vectors that we will be discussing shortly in section 6.2.

### 3.1.1 File-matching (SHA-1)

```
<dict>
  <key>MatchType</key>
  <string>Match</string>
  <key>Identity</key>
  <data>eX17YAgTaOUMt9icXVHF0meoiog=</data>
  <key>MatchFile</key>
  <dict>
    <key>NSURLNameKey</key>
    <string>RBSHell.rbx_0.129.dylib</string>
    <key>NSURLTypeIdentifierKey</key>
    <string>com.apple.mach-o-dylib</string>
  </dict>
</dict>
```

In the above snippet, we can see that it's for a specifically named file, with a specific type (i.e. com.apple.mach-o-dylib), and a sha1 hash of the file.

```
$ echo eX17YAgTaOUMt9icXVHF0meoiog= | openssl base64 -d | hexdump
00000000 79 7d 7b 60 08 13 68 e5 0c b7 d8 9c 5d 51 c5 d2
00000010 67 a8 8a 88
00000014
```

As a comparison, here is the sha1 hash of file distributed with the version of HellRaiser we are looking at:

```
$ openssl dgst -sha1
malware.app/Contents/Frameworks/RBSHell.rbx_0.129.dylib

SHA1(malware.app/Contents/Frameworks/RBSHell.rbx_0.129.dylib)=
797d7b60081368e50cb7d89c5d51c5d267a88a88
```

# APPLE MACOSX AND MALWARE – MYTH VS REALITY

## 3.1.2 File-content matching (strings within the file)

Some of the other entries also look interesting:

```
<dict>
  <key>MatchType</key>
  <string>Match</string>
  <key>Pattern</key>
  <string>656C6C5261697365722053657276657200165F44454255475F4C4F475F505249564154452E747874*
5374617274536572766572203E20212053455256455220524553544152544544*2F7573722F62696E2F646566661756C7473207772
697465206C6F67696E77696E646F77204175746F4C61756E636865644170706C69636174696F6E44696374696F6E617279202D617
27261792D61646420273C646963743E3C6B65793E486964653C2F6B65793E3C00192F3E3C6B65793E506174683C2F6B65793E3C73
7472696E673E00113C2F737472696E673E3C2F646963743E27*48656C6C52616973657220536572766572</string>
  <key>MatchFile</key>
  <dict>
    <key>NSURLTypeIdentifierKey</key>
    <string>{dyn.*}</string>
  </dict>
</dict>
```

Firstly, the “NSURLTypeIdentifierKey” is the Uniform Type Identifier (UTI) key. UTI's can cover anything from paste board data to folders and files, and the “dyn” tag says that it's an unknown (or dynamic) type, so it doesn't really narrow it down much. The UTI for the sample I have is “dyn.ah62d4rv4ge8xe”. The advantage of this type of UTI is that even if the binary is run directly (rather than the .app bundle), it will still match.

The pattern looks interesting. First, let's try to get a string out of it.

```
$ perl -le 'print pack("H*",
"656C6C5261697365722053657276657200165F44454255475F4C4F475F505249564
154452E747874");'
```

```
ellRaiser Server_DEBUG_LOG_PRIVATE.txt
```

That looks good, so we process the entire string, and once we have decoded it, it looks like this (I've broken the lines at the \*s for readability):

```
ellRaiser Server_DEBUG_LOG_PRIVATE.txt*
StartServer > ! SERVER RESTARTED*
/usr/bin/defaults write loginwindow
AutoLaunchedApplicationDictionary -array-add
'<dict><key>Hide</key></><key>Path</key><string></string></dict> '*
HellRaiser Server
```

All these strings exist in the binary, and after some testing it appears that unless they all exist, no match occurs. This check may be as simple as comparing the output of the strings command against these strings. Changing the strings in the binary avoids a match based on this rule.

Interestingly, if more than one binary exist in the Contents/MacOS directory of the .app bundle, they are also used for matching.

# APPLE MACOSX AND MALWARE – MYTH VS REALITY

## 3.1.3 Overall matching

A single entry in the top-level dictionary defining some malware is made up of an array of “Matches”, all of which must be true before an item is flagged as malware. In the example of HellRaiser, the first entry “OSX.HellIRTS” is defined as having “rbframework.dylib”, “RBSHell.rbx\_0.129.dylib” (which we examined above) and certain strings found in the binary (that we also looked at above). It’s valid to have multiple entries with the same name. So “OSX.HellIRTS” also has a further two entries which just use hashes on the binary itself.

## 3.1.4 Other matching types

The match types discussed are not exhaustive, and others may exist.

## 4 The good

Having malware protection built into the OS is a great feature, and Apple should be praised for providing some protection.

Using UTIs as a method of matching malware that has some potential, and the ability to build up fairly complex rules should provide a lot of flexibility. This is assuming, of course, that the definition file is updated with some regularity. Apple’s built-in application firewall also works well, and adds an extra level of protection.

In fact, we suggest that the firewall should always be enabled, and “Enable access for assistive devices” should be disabled (with the reasons for this being outside the scope of this document, but is something we might look at a later date).

## 5 The bad

It’s not all roses.

First of all, only applications with “LSFileQuarantineEnabled” set to true will create files marked with “com.apple.quarantine”, and hence files acquired via other means will not have any protection.

Another way of looking at this statement is that the application itself can choose whether the “LSFileQuarantineEnabled” bit is set and therefore whether all files they download are quarantined and checked by XProtect.

So the obvious question is:

### 5.1 What programs have LSFileQuarantineEnabled set?

Based on average systems, it appears all common web browsers and email clients do have this set, either in the applications plist or in the system-wide “Exception.plist”, however, IM and, worryingly, P2P programs tend not to. Apple’s own programs are usually well behaved. Both Mail and iChat will create quarantined files.

The trouble though is that Instant Messenger and especially P2P programs are designed to exchange files. That is actually their whole purpose!

# APPLE MACOSX AND MALWARE – MYTH VS REALITY

You can have Mac OS X up to date, but if you download a Mac OS X targeting Trojan with your torrent network, XProtect will not protect you! And downloading files is the only reason why people use torrent clients obviously.

If some malware does manage to run on a system, it can freely download and execute whatever it likes with no interruption from the OS.

## 5.2 Some attack vectors

We have established that XProtect is only invoked for quarantined files, and this leaves obvious attack vectors.

The first one is that when applications are executed by other applications, the checking doesn't seem to happen. For example, a signed Java applet can download and execute a program, and, although the downloaded binary is correctly flagged with "com.apple.quarantine", no special prompting occurs.

This binary can create unflagged binaries and trick the user into executing them. The assumption is that only if the binary is launched with "LaunchServices" (which all .app bundles will be) will any checking happen. So, if a helper application does a fork()/exec(), no checking will occur.

Any blacklist scheme is faced with the problem of false positives, and XProtect is no different. So, the reasonable thing to do is provide the user with a choice, which is exactly what happens. However, if Windows is any example, it proves that some people will ignore whatever warnings are given and still click to proceed.

In a similar light, there are programs such as keyloggers, which are sold as (somewhat) legitimate products. Apple's malware defences will not provide any protection against such programs.

## 5.3 Performance considerations

The question of performance is also interesting.

*"The number of new malicious programs detected during the year was approximately 13 million" - Kaspersky<sup>2</sup>, 2010*

While the model of only running a scan on files flagged for quarantine means the performance impact should only hit on the first run of a downloaded application, the question of performance and scalability has to be asked. If the situation on Windows is anything to go by, Apple may have to deal with very large numbers.

Currently, "XProtect.plist" only has three different groups of malware. If we include variants we might be generous and say it detects 50 different strains. What happens if this number is increased?

I added 10,000 entries, each with a pattern match field to "XProtect.plist", and it added about 7 seconds (!) to the time it took to launch a file flagged for quarantine that didn't match any rule. This

---

<sup>2</sup> Taken from [http://www.kaspersky.com/reading\\_room?chapter=207717661](http://www.kaspersky.com/reading_room?chapter=207717661) - Malware Evolution 2010

# APPLE MACOSX AND MALWARE – MYTH VS REALITY

last clause is important, because “CoreServices” is smart enough to stop searching as soon as it finds a match. So the extra size didn’t make finding malware early in the list any slower.

This, in effect, means that non-malware faces an aggravating time penalty if the “XProtect.plist” gets too large.

## 6 Conclusion

It will be of no surprise to anyone reading this document that the idea of Mac OS X having perfect security is a complete myth. There is, however, a grain of truth in the idea that there is no malware on Mac OS X, albeit a small grain that is shrinking rapidly! As it stands today, there isn’t much malware that we can find.

That RATs (Remote Administration Tools) such as Sophos’s recently discovered Black Hole and HellRaiser get publicity says something about the current quantity and state of malware on Mac OS X. However, this shouldn’t be an excuse for Mac users to be lulled into a false sense of security. The malware is out there, and it’s growing both in terms of numbers and sophistication.

Apple is being proactive and introducing OS level security. It’s flawed security that is currently pretty flimsy, but nonetheless, it’s a (small) step in the right direction. It’s going to be interesting to see the direction Apple chooses from here.

iOS will only run signed binaries, so this type of protection is less important for that OS. Will XProtect get improved? Or will Mac OS X get locked down and only run signed binaries? And, how long will it be until there is a serious outbreak of malware on Apple’s platforms?