Engineering Development Group

ELSA
v.1.1.0
User Manual

Rev. 1.0.0
September 27, 2013

Change Log

Doc Date	Change Description	Authority
June 13, 2012	Initial Draft	2277218
June 18, 2012	Classification banner, additional data on csv format, background on wifi geolocation, documented error (base64) behavior	5086947
July 3, 2012	Specific install instructions for each mode, reorganized headings, added troubleshooting guide	5086947
July 17, 2012	Added instructions as a result of 1st operational deployment.	2277218
July 18, 2012	Added notes on service groups, backoff factor, service dependencies and log file deletion as a result of 1st operational deployment.	5086947

Table of Contents

List of Figures

1. (U) Scope

(S) This document establishes the User Manual for ELSA v1.0.

1.1 (U) System Overview and Description

(S) ELSA is a software system that geolocates wifi-enabled computers. Elsa provides pattern of life geolocation information by recording the details of wifi access points near the target machine and transmitting that metadata to 3rd Party databases for resolution into latitude, longitude and an accuracy measure. These 3rd party databases exist to support location services in the Firefox, Chrome and Internet Explorer browsers according to the w3c specification. ELSA uses HTTPS connections to query these 3rd party services and saves its data into a 128 bit AES encrypted file.

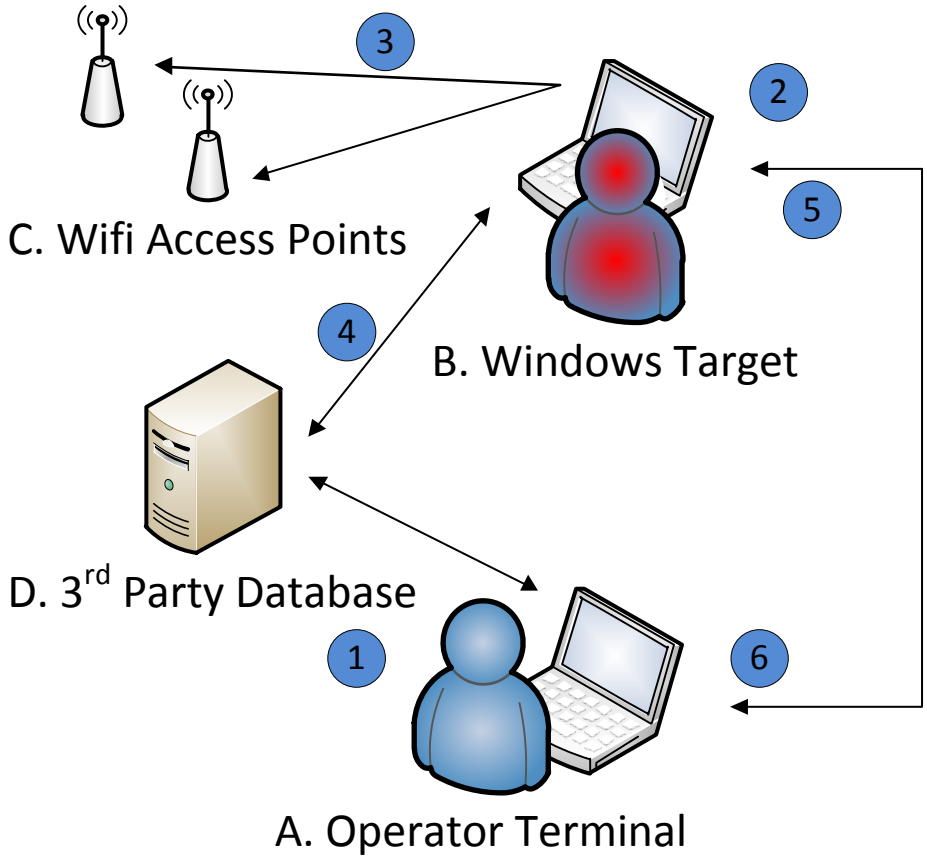

Figure 1 - (S) Operational scenario for Elsa

(S) See Figure 1 above for a typical operational scenario. As shown, the Elsa software system consists of two major components. First, the processing component (A. Operator Terminal) will typically reside on an operator's ICON attack box. Second, the implant (B. Windows Target) will typically be deployed on a target Windows host. Elsa can either directly resolve locations from the 3rd party database (D. 3rd Party Database), or optionally return unresolved wifi data to the operator terminal so that it can be resolved from there.

(S) In order for Elsa to successfully geolocate a target, the target must be (1) wifi enabled, (2) deployed in an environment with wifi access points in range (C. Wifi Access Points) and (3) deployed in an environment where a survey has recently been performed by the provider.

(S) Given these criteria are met, Figure 1 above illustrates typical Elsa operation on a Windows client:

1. An operator configures an Elsa implant based on the host environment.
2. An operator deploys the implant to a Windows host and it begins collection.
3. The implant begins collecting wifi access point information based on the schedule set by the operator.
4. If configured to do so, when the target user connects to the Internet the implant will resolve the wifi data to a geolocation via the 3rd party database.
5. The operator connects to the Windows hosts and downloads the encrypted collection log.
6. The operator decrypts the log and performs further analysis on their target. The operator can use the wifi data to query geolocation information from alternate databases.

(S) Although the implant is the recommended means for collecting wifi metadata from targets, the processor can ingest and resolve wifi metadata obtained through other means so long as the MAC address, SSID and signal strength are available. See the processor instructions for more details on the comma-separated-value feature.

1.2 (U) Assumptions and Constraints

(S) As long as the wifi is enabled, the implant can record wifi observations. This is true even if Windows host is not actually connected to an access point. If configured, during each geolocation interval the implant will attempt to resolve geolocation information for past observations. A connection to the Internet is required for this to occur.

(S) Wifi data points are larger in size than geolocation coordinates so the tool provides settings for limiting the log size. When the log size is reached the implant deletes older data points in favor of newer ones. The log is encrypted and written to disk to persist the collection across reboots. By default, when the tool resolves a wifi observation to a geolocation coordinate it will delete the wifi observation.

1.3 (S) Background on Wifi-based Geolocation

(U) A wifi geolocation provider first amasses a database containing the geolocations of many wifi access points. This data can be collected by any device with (1) a wifi receiver and (2) knowledge of its true location; for example, specially outfitted cars or typical smartphones. Once built, this database associating wifi metadata (MAC addresses, signal strengths and sometimes SSIDs) with locations can be queried to resolve the metadata to a location. Providers can use the relative signal strengths observed by the machine to better triangulate the machine between multiple access points.

(U) Wifi-based geolocation has the advantage that it does not require a GPS or cellular chip on the device - only a common wifi interface. However, geolocation is at the mercy of the quality of the database. If the database isn't up to date over the entire geographic region of interest then geolocation results will be poor.

(U) A simple test to determine whether a region has good wifi-based geolocation coverage is as follows:

1. In the region of interest, open a recent version of the Chrome or Firefox browser on a machine with a wifi interface and Internet connection
2. Navigate to maps.google.com
3. Click on the 'locate me' button above the zoom slider
4. If prompted, allow the browser to access your location
5. If the map accurately displays your location, then coverage is good

(U) It is worth noting that under the w3c geolocation API the web browser, not the site, is responsible for resolving geolocations. If you click 'locate me' in google maps on IE9 , a request will be sent to Microsoft's servers, not Google's. At the time of writing Firefox, Chrome and Opera all use Google's database.

2. (U) Applicable Documents

(U) The following documents, of the exact issue shown, form a part of this document to the extent specified herein. In the event of a conflict between the documents referenced herein and the contents of this document, the contents of this document will be considered binding. The following documents may be found in the EDG PMB database:

- Requirement 2012-0565.doc

3. (U) System Description

(S) As described above, Elsa is a software system designed to provide COG pattern of life geolocation information. The major component is the Windows DLL (tool), which is used in the target environment. The minor components are the configuration tool (patcher) and the post processor (processor), which are used in deployment, configuration, and operation.

(S) The following steps describe the typical usage of Elsa.

1. Assess the target for the desired frequency of geolocation data points based on the likely operational access opportunities.
2. Use the patcher to set the geolocation interval and data store file size.
3. Deploy the tool.
4. Recontact the target and download the collection log.
5. Run the post processor to decrypt the log for analysis.

3.1 Technical References

Directory	Filename	Note(s)
server/windows	patcher.exe	Windows Config Tool
server/windows	processor.exe	Windows Decryption Tool
server/windows	tool-x64.dll	Windows x64 implant
server/windows	tool-x86.dll	Windows x86 implant
server/windows	installDllMain.vbs	VBScript file illustrating optional placement of the Elsa task in the Task Scheduler
server/windows	uninstallDllMain.vbs	VBScript file illustrating optional removal of the Elsa task from the Task Scheduler
server/windows	sha1-windows-images.txt	Sha1 hashes of files in the distribution
server/windows	classifications-windows.txt	Classifications of files in the distribution
docs	Elsa User Manual.pdf	This manual

Figure 2 - (S) Listing of files included with the Elsa distribution

3.2 (U) System Concepts and Capabilities

(S) An operations officer must have a broad understanding of the Windows command line interfaces, wired and wireless computer networking, and Windows system administration. The officer must be familiar with Asset, Supply Chain, or Remote Operations tools and procedures.

3.3 (S) DLL Injection

(S) The Elsa client is designed to be injected into an existing process on the system. It is delivered in the form of a DLL. As such it is important that the 32 bit and 64 bit versions of the DLL be run on the matching 32 bit and 64 bit version of Windows.

(S) Some Anti-Virus (AV) suites protect critical system processes such as SERVICES.EXE and WINLOGON.EXE from the dll injection technique used in ELSA

but allow injection for non-critical and 3rd party (non-Windows) processes. Deploying ELSA to these systems require careful system survey, targeting, and/ or cover application for processes vulnerable to this type of injection.

(S) ELSA is able to run in several operational modes that offer flexibility as to its appearance within the system. ELSA can be installed as a service running inside of SvcHost, a scheduler task running inside of DllHost, a utility running inside of rundll32, or as an AppInit Dll running inside of a specified process.

3.4 (U) Dll Installation

(S) The ELSA dll exports routines that can be used to install the tool using RegSvr32.exe among other methods (see 'Deployment' below). In some AV environments these installation routines can result in pop up messages. Signing the tool with certificates can mitigate these messages. As an injectable dll the injection can also be used for the installation process. See the InstallFromProcess section for more information on this procedure.

(S) The 64 bit version of Windows contains two copies of RegSvr32.exe. The first is a 64 bit executable located in the C:\Windows\System32 directory and the second is a 32 bit executable located in the C:\Windows\SysWOW64 directory. Deploying the 64 bit version of the dll requires the 64 bit version of RegSvr32.

3.5 (U) Prerequisites

(S) The software system will only operate correctly if each component is executed within the appropriate operating environment.

3.5.1 (U) Operator Terminal Requirements

(S) The operator terminal is designed to run on Windows 7.

3.5.2 (U) ELSA Requirements

(S) The ELSA implant may be executed on any of the following versions of Microsoft Windows:

1. Windows 7 32 bit, 64 bit

3.6 (U) Equipment Familiarization

(S) An operations officer must have a broad understanding of the Windows command line interfaces. They must understand the configuration and installation of Windows services, schedule tasks, and drivers.

3.7 (U) Security Provisions

(S) The ELSA tool can be renamed, signed, and packed without losing its functionality. The ELSA dll contains no strings that would give away its true purpose. The dll and driver do not hide their processes or files. The dll can be injected inside an existing process which reduces its visibility. The ELSA data file back dates its timestamp to its original creation date each time the file is updated.

4. (U) Preparation

(S) Prior to deciding to use ELSA it is worth making an assessment of how likely the target machine is to (1) have wifi turned on, (2) be near wifi access points and (3) be in a region where Google or Microsoft have high quality wifi databases. All three elements are required for Elsa to succeed. See the above 'Background on Wifi-based Geolocation' for suggestions on evaluating the third item.

(S) Operation of ELSA involves using the PATCHER wizard to create an ELSA implant. Once ELSA is deployed and running the data file can be collected. The PROCESSOR is then used to decrypt the data file. The PROCESSOR can then be used to resolve any unresolved access point lists contained in the decrypted data file.

(S) The ELSA software system is delivered in two sets of zip files with embedded hash files containing the project name, version, and algorithm used to calculate the hash:

- elsa-v1.0.0-docs.zip
  - sha1-windows.txt
- elsa-v1.0.0-windows.zip
  - sha1-windows-images.txt

(S) After extracting the files from the zipped archives the file hashes should be verified against the hashes in the distribution to verify that the media has not been corrupted. The project version printed in the hash file should also be checked to validate that distribution is the current version approved for operational use. Note that the hashes in these text files should be used in favor of the hashes in this manual, which may be out of date.

```
file: sha1-windows-images.txt
project: elsa
version: 1.0.0
date: 2012-06-14 13:41:04
description: elsa hashes calculated using the SHA1 algorithm.

patcher.exe: 9b416af5178830a79f07f68b10f6ea7b8c18a7c0
tool-x64.dll: 250e9e11a5416f1fa477c2736e54bcf3c3b8b202
tool-x86.dll: 2914b324b926b1f9fdf854749a7d6df169773d4b
processor.exe: 313babc073b6d7981649f985f54d6ea7a7a11c6e
addtask.vbs: c1af8b9f6191236c8bc69f04e01dfc7d84b4fa5e
```

Figure 3 - (S) SHA1 Hash File Format

(S) At this point the user is ready to configure Elsa for deployment using the PATCHER tool. The user will need to be prepared to specify:

1) The target machine's architecture (x86 vs. x64)
2) The desired mode (dllhost, svchost, rundll32 or appinit – See *Deployment* below)
3) The desired geo provider (microsoft / google)

4) The desired maximum log file size
5) Whether or not to resolve ap lists into geos from the target

4.1 (U) PATCHER Configuration Tool

(S) The PATCHER tool generates a deployable Elsa implant (dll) based upon the configuration options specified by the user. These configuration options can be specified in a file or interactively in wizard mode. The general PATCHER command line syntax is as follows:

```
> patcher.exe –p [x86|x64] –o OUTPUT_FILE [–c CONFIG_FILE] [-W]
```

(S) The specific syntax for wizard mode and config file mode are as follows:

```
> patcher.exe -p [x64|x86] –o new_tool.dll –W
- or –
> patcher.exe –p [x64|x86] –o new_tool.dll –c new_tool.config
```

Figure 4 - (S) Command line syntax for the PATCHER tool

4.1.1 (U) PATCHER option Processor Architecture

(S) The processor architecture parameters specifies the creation of either the x86 or x64 version of the dll. For system services such as SvcHost, DllHost, and RunDll32 the processor architecture must match. The switch for this parameter is '-p'.

4.1.2 (U) PATCHER option Output File

(S) The output file option specifies the output file name for the ELSA implant. The switch for this option is '-o'.

4.1.3 (U) PATCHER option Config File

(S) The config file option specifies the ELSA configuration file to use when creating an implant. This option does not need to be specified, if the operator is using the Wizard mode to generate a configured implant. The switch for this option is '-c'.

```
Mode = RunDll32
TargetProcess = rundll32.exe
DataFileName = %SystemRoot%\TEMP\elsag.data
DataFileMaximumSizeKB = 202
DataFileArchiveSeconds = 62
DataFileKey = F4CD1BC482E98849027CACB150FB96247E60A2CCA329114167DB0710FA679823
Guid = {59553112-3228-49ce-8044-4AB3C63BD46C}
WifiSurveyIntervalSeconds = 32
WifiSurveyInstallDelaySeconds = 32
WifiSurveyStartupDelaySeconds = 32
WifiSurveyFailureBackoffMultiple = 13
WifiRssiThreshold = 303
WifiSaveAllSurveys = false
GeoProvider = google
ClientID = 2234
```

Figure 5 - (S) Example wizard.config file settings.

4.1.4 (U) PATCHER option Wizard

(S) Specifying the -W flag will run the PATCHER in wizard mode, interactively prompting the user to specify ELSA settings. Upon completion PATCHER will generate a basic ELSA config file named 'wizard.config' that contains the settings specified by the operator.

(S) The 'wizard.config' file can be modified for more advanced configuration scenarios. The config file format supports UTF-8 allowing for foreign language characters in the service description fields and file paths. The wizard prompts do not support foreign language input thus the file must be edited manually to do this.

```
> cd unclassified\server\windows
> patcher.exe –p x64 –o testx64g.dll –W
Enter mode [default is RunDll32]:
Enter target process filename [default is rundll32.exe]:
Enter data file name [default is %SystemRoot%\TEMP\elsa.data]:
%SystemRoot%\TEMP\elsag.data
Enter data file max kb [default is 200]: 202
Enter data file archive seconds [default is 60]: 62
Enter data encryption key file [default is key.bin]:
Create a new application guid [default is no]:
Enter application guid [default is {59553112-3228-49ce-8044-4AB3C63BD46C}]:
Enter seconds between wifi surveys [default is 30]: 32
Enter the seconds to delay the wifi survey after install [default is 30]: 32
Enter the seconds to delay the wifi survey after startup [default is 30]: 32
Enter backoff factor to use when wifi survey are unsuccessful [default is 10]: 13
Enter the wifi rssi threshold [default is 100]: 202
Save all wifi surveys [default is no]:
Enter geolocation provider [default is google]:
Enter the Client Id [default is 5555]: 2234
```

Figure 6 - (S) Example command line usage of the PATCHER wizard

4.2 (U) Elsa CONFIG Option Details

(S) Elsa's behavior is defined by the configuration options provided to the patcher, so it is important to understand these options and carefully consider the desired behavior prior to deployment. A few significant options to consider are:

1) Do you want Elsa to try to resolve geolocations from the target, or would you prefer to resolve them later using the processor tool? (See GeoProvider option)
    - Resolving from the target creates additional network traffic, although this traffic is designed to look like legitimate browser traffic
    - Resolving later could produce different results if the provider's wifi database changes, although this is probably not likely

2) How much space are you willing to allocate for the log file? (See DataFileMaximumSizeKB option)

3) Do you want to save the wifi surveys after they've been geolocated? (See WifiSaveAllSurveys option)
   - Wifi surveys may get deleted even if geolocation fails – resulting in a loss of data
   - Wifi surveys can get large if the target machine is in an area with a lot of access points nearby

(U) The following sections provide detailed descriptions of each configuration option.

4.2.1 (U) CONFIG option DataFileName

(S) The DataFileName is the location of the encrypted collection file on the target system. ELSA will expand environment variables but the operator should ensure that the environment variable exist for the account. The accounts used to uninstall ELSA can be different than the account that ELSA runs as, although in that case care must be taken to ensure that variation in environment variable expansion does not leave data on the machine.

(S) For example, consider the case where ELSA is installed from the Administrator account and configured to run as a system service. If %TEMP% is specified as the file path then during the install (AND uninstall) %TEMP% will expand to 'C:\Document and Settings\Administrator\Local Settings\Temp'. When ELSA starts up as the system service %TEMP% will expand to 'C:\Windows\Temp' e.g. the value specified in the HKLM\CCS\Control\Session Manager\Environment registry setting. When the uninstall procedure is run from the Administrator account it will appear as though ELSA is not deleting the data file. Depending on the system configuration it may be preferable to either specify %SystemRoot% or hard code the path.

4.2.2 (U) CONFIG option DataFileMaximumSizeKB

(S) The DataFileMaximumSizeKB option specifies the maximum size to which the data file is allowed to grow. It is not an exact size as encryption adds some overhead. This size is used to compute the records to save.

(S) The size of wifi records vary depending on the density wifi access points within the operating environment. On average a single wifi access point will require 44 bytes of storage and a location with 20 access points within range will require 900 bytes of storage. There is currently no limit on the number of access points recorded in a wifi record, so dense environments can result in arbitrarily large data files. Geolocation coordinate records are smaller and the size of a single coordinate averages about 52 bytes.

(S) ELSA will bias the collection to preserve geolocation coordinates over wifi access point observations. Once out of space ELSA will drop the oldest wifi observations first followed by the oldest geolocation coordinates once all wifi observations are gone. As

described below it is important to note that in this mode the current version of Elsa will delete ap lists even if it encounters errors in parsing geolocation responses. This can result in data being lost.

(S) When ELSA observes a set of access points it compares the new observation to the last observation. If they differ then the new set of access points is used to request a location; otherwise, it is discarded since an additional request would presumably result in an identical location. ELSA uses the following criteria to determine if two access point lists differ:

- If the old wifi list is empty then they differ
- If the lists do not contain the *exact same* set of MACs and BSSIDs, then they differ
- If the lists contain the exact same set of MACs and BSSIDs but the largest signal strength difference between two corresponding APs is greater than the WifiRssiThreshold option then they differ
- Otherwise the lists are considered the same

(S) Note that the above criteria were designed to be highly sensitive to change and as a result can result in a lot of geolocation requests being sent. If essentially *anything* changes about the lists then a new record is logged and a new request sent (if elsa is configured to request geos). Testing has revealed that lists can change quite frequently even when a device is motionless, so these criteria may be revisited in the future.

4.2.3 (U) CONFIG option DataFileArchiveSeconds

(S) The DataFileArchiveSeconds option specifies the number of seconds between updates to the disk.

4.2.4 (U) CONFIG option DataFileKey

(S) The DataFileKey option is the hex encoded 128 bit AES encryption key used to encrypt the data file. This key will be needed for use in the PROCESSOR to decrypt the data file. The PATCHER wizard asks for the filename that contains the key. The default filename is 'key.bin'. If the file does not exist then the wizard prompts if the file should be created.

4.2.5 (U) CONFIG option Mode

(S) The Mode option specifies the operational mode of ELSA and is covered in detail in the deployment section. It is needed for ELSA to operate as well as for the ELSA install and uninstall routines.

Mode	Description
SvcHost	Enables the Service entry point.
DllHost	Enables the Task Scheduler entry point.
RunDll32	Enables the RunDll32 entry point.
AppInit	Enables the AppInit entry point.

Figure 7 - (S) Elsa Mode Settings

4.2.6 (U) CONFIG option TargetProcess

(S) The TargetProcess option specifies the module name in which the ELSA dll should execute. It should be set to match the Mode setting and the PATCHER wizard helps set this up correctly. The setting is not case sensitive. In AppInit, mode ELSA can be configured to run inside any particular exe as long as LoadLibrary() is called to load ELSA. When using AppInit mode the TargetProcess should match the file name of the exe on the target system e.g. renaming matters.

TargetProcess	Description
SvcHost.exe	Setting when SvcHost mode is used.
DllHost.exe	Setting when DllHost (Task Scheduler) mode is used.
RunDll32.exe	Setting when RunDll32 mode is used.
Explorer.exe	An example setting for AppInit mode. It could be almost any valid application such as Calc.exe or Notepad.exe

Figure 8 - (S) Elsa Target Process Settings.

4.2.7 (U) CONFIG option InstallFromProcess

(S) The InstallFromProcess option is used for alternate installation procedures when RegSvr32 is not viable. As discussed in the *Deployment* section, RegSvr32 is used to load Elsa in the SvcHost and DllHost (Task Scheduler) modes. When the dll is loaded if the InstallFromProcess setting matches the module name then Elsa can be instructed to install itself onto the system. This will require the creation of a signal file in the same directory as the dll. The signal file is a file name with the same name as the dll but with the '.install' file extension. Elsa can also be uninstalled in the same fashion by using the '.uninstall' extension.

Filename	Size	Description
C:\Windows\System32\elsa.dll	100 KB	ELSA dll
C:\Windows\System32\elsa.install	0	ELSA install signal file

Figure 9 - (S) Elsa Alternate Install Procedure

4.2.8 (U) CONFIG option ServiceName

(S) The ServiceName option is used for the SvcHost mode. It specifies the service registry key name.

4.2.9 (U) CONFIG option ServiceGroup

(S) The ServiceGroup option is used for the SvcHost mode. It specifies the SvcHost group in which to run ELSA. When choosing a ServiceGroup it may be helpful to view the list of existing service groups in the following registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost

4.2.10 (U) CONFIG option ServiceDisplayName

(S) The ServiceDisplayName option is used for the SvcHost mode. It specifies the service display name for the services.msc control panel applet.

4.2.11 (U) CONFIG option ServiceDescription

(S) The ServiceDescription option is used for the SvcHost mode. It specifies the description field for the services.msc control panel applet.

4.2.12 (U) CONFIG option Guid

(S) The Guid option is used whenever the implant needs a guid. As a dll the tool needs a mechanism for detecting multiple instances of the implant. Elsa creates a global named object using the guid value to discern previously running instances. Additionally, when installed as a scheduled task, DllHost mode, Elsa needs a CLSID with which to register on the system so Elsa uses the guid value for the CLSID.

4.2.13 (U) CONFIG option WifiSurveyIntervalSeconds

(S) The *ISECONDS* parameter is the number of seconds between wifi geolocation periods. The units are in seconds and is required. The interval is measured from the last time to tool was able to obtain geolocation coordinates.

4.2.14 (U) CONFIG option WifiSurveyInstallDelaySeconds

(S) The *DSECONDS* parameter is the number of seconds to delay after install. It is calculated from the dll file creation time. The option is in seconds and is required.

4.2.15 (U) CONFIG option WifiSurveyStartupDelaySeconds

(S) The *SSECONDS* parameter is the number of seconds to delay after startup. It is calculate from the time the client is loaded. Using RunDll invocation the start time is from when the command is issued. Using AppInit invocation the start time is typically the machine boot for a system process. The option is in seconds and is required.

4.2.16 (U) CONFIG option WifiSurveyFailureBackoffMultiple

(S) The *BMULTIPLE* parameter is used when the client fails to receive an HTTPS response during geolocation attempt. The multiple is used adjust the interval to reduce the geolocation frequency and prevent frequent failed network connections. The calculation uses integer math and the default value is 10, which tells the system to re-use the original configured interval.

New Survey Interval Seconds = *ISECONDS* * *BMULTIPLE* / 10

Figure 10 - (S) PATCHER Backoff Calculation

(S) Note that this parameter will not be used if the client receives ANY HTTPS response whatsoever, even if this response does not contain a geo. See the references to captive portals in the 'Interpreting Errors in ELSA xml files' section.

4.2.17 (U) CONFIG option WifiRssiThreshold

(S) The *ITHRESHOLD* parameter is used during the decision to record a new data point. If the wifi observations BSSIDs and MAC address are the same then this parameter determines, if the new observation will be recorded. For a matching access point sets the algoritm identifies at the largest magnitude difference between the signal strengths, if it is greater than *ITHRESHOLD* the record is saved.

4.2.18 (U) CONFIG option WifiSaveAllSurveys

(S) This option controls the wifi observation storage behavior. The default operation of Elsa is to delete wifi observations when a geolocation coordinate is obtained by the location provider. This is a space saving measure. Setting this option to true will result in Elsa saving all wifi observations until the DataFileMaximumSizeKB limit is reached.

4.2.19 (U) CONFIG option GeoProvider

(S) This option specifies the 3rd party database to use when resolving wifi observations. The valid settings are as follows:

Setting	Description
none	This disables the geolocation query in the tool. This can be useful if the tool must be installed on a machine with an aggressive AV or aggressive network defense team. The collection file would only contain the wifi observations and the operator uses the PROCESSOR to query the 3rd party geolocation database.
google	This settings configures the tool to query google location services for geolocation coordinates.
microsoft	This setting configures the tool to query microsoft location services for geolocation coordinates.

Figure 11 - (S) Elsa location provider settings

4.2.20 (U) CONFIG option ClientID

(S) The *CLIENTID* parameter should be a four-digit unsigned hexadecimal number that will serve as the unique, operationally assigned ID number for the client. The default value is 5555 and is required.

5. (U) Deployment

(S) As described in the config settings, ELSA supports several running modes, each with slightly different installation procedures. In all modes the configured dll generated by the PATCHER must first be copied onto the target machine.

Mode:	Description:	Installation Overview:
SvcHost	Runs the dll as a service, visible in the Services panel	RegSvr32 and net start commands
DllHost	Runs the dll as a scheduled task, visible in the Task Scheduler	RegSvr32 and an install script
RunDll32	A general tool for loading and running dlls.	RunDll32 command – this is not really an 'installation' and will not persist
AppInit	Add Elsa to a list of dlls that load when *any* app runs.	Edit registry keys

Figure 12 - (S) Overview of Elsa Modes and Installation

(S) During deployment it is often helpful to use the tasklist command to see if the dll is loaded and running somewhere. You can also use asterisks as wildcards in the dll name argument to double check that you don't have any old versions of the dll loaded:

```
> tasklist /m <dll name>
```

(S) Elsa uses the Microsoft Windows RegSvr32 tool to perform the installation; however, this tool will not install if the dll is placed in the C:\Windows\system32 directory. The Elsa dll can be installed from other directories such as the 'C:\Windows' or the 'C:\Program Files' directories.

(S) General practice is for 64 bit dlls to be deployed to 64 bit systems; however, if an installation host such as 32 bit Taper is used then the operator will want to be aware that the 64 bit version of Windows contains two copies of RegSvr32.exe. The first is a 64 bit executable located in the C:\Windows\System32 directory and the second is a 32 bit executable located in the C:\Windows\SysWOW64 directory. Deploying the 64 bit version of the dll requires the 64 bit version of RegSvr32.

(S) The following command line syntax can be used to install x64 bit elsa dlls from a 32 bit process:

```
> %WINDIR%\sysnative\regsvr32.exe /s %WINDIR%\ELSA_x64.dll
```

5.1 (S) SvcHost Mode Installation

(S) This mode will add Elsa as a windows service, running within one of the svchost.exe processes on the system. Installation proceeds as follows:

1) Run the patcher with the appropriate processor architecture specified:
    1. Select SvcHost as the Mode
    2. Select SvcHost.exe as the target process filename
    3. Enter a service name, display name, description and service group – **these will be visible to the target user**
    4. Complete the remaining patcher options
2) Place the patched Elsa dll on the target system
3) Check that the wlansvc and eaphost services are running as follows (see troubleshooting section if they are not running):

```
> sc query wlansvc
> sc query eaphost
```

4) Substituting the appropriate dll name, silently load the dll using the RegSvr32 utility:

```
> RegSvr32 /s C:\ELSA.DLL
```

5) You can now verify that the display name (e.g. - "Windows Management Instrumentation Device Extensions") in the services panel.
6) The service is just registered in the Services panel – it is not running. It will automatically start on reboot, but if you want to run it immediately use the following command with the appropriate service name (not display name):

```
> net start wmidx
-or-
> sc start wmidx
```

7) The service will restart automatically on reboot.

5.2 (S) DllHost/Task Scheduler Mode Installation

(S) Dllhost mode is highly analogous to svchost mode; the dll must be registered and then a Windows system process must be instructed to load it. In this case the Windows Task Scheduler loads the dll as a 'task'. This will only work on systems with Task Scheduler 2.0 or greater.

(S) Since creating the task scheduler entry is more complex than starting the service in SvcHost mode, a script has been provided (see "Scripts for DllHost Mode Install / Uninstall"). The operator can either manually run this script or let Elsa run it automatically when it is registered using the RegSvr32 utility. This script is just an

example and requires customization by the operator. For example, it uses WScript.Echo messages that the operator may wish to comment out or remove.

1) Run the patcher with the appropriate processor architecture specified:
    1. Select DllHost as the Mode
    2. Select dllhost.exe as the target process filename
    3. Complete the remaining patcher options
2) Rename the install script <dll name without '.dll' extension>.install.vbs
3) Edit the install script to set Action.ClassId equal to the GUID used in the patcher
4) Edit the install script task schedule options as desired - **these options will be visible to the target user in the Task Scheduler**
5) Make sure the edited install script runs by double clicking it on a local machine. A Task should appear in your local Task Scheduler, which you can delete
6) Place the patched dll and the install script on the target machine in the same directory
7) Substituting the patched dll's name and path, silently register the dll using the RegSvr32 utility:

```
> RegSvr32 /s C:\ELSA.DLL
```

8) You should see the install script disappear
9) Check to see that the task has been scheduled using the schtask utility and the selected test name:

```
> schtasks /query /tn "Test Daily Trigger"
```

10) Elsa will load and run at the time you've prescribed in the install script

5.3 (S) RunDll32 Mode Installation

(S) RunDll32 is the simplest mode in which to run Elsa. Unlike SvcHost and DllHost, no system process like the Services panel or Windows Task Scheduler is configured to load Elsa automatically on startup. This means that the operator will need to use some other tool to restart Elsa across reboots.

1) Run the patcher with the appropriate processor architecture specified:
    1. Select RunDll32 as the Mode
    2. Select rundll32.exe as the target process filename
    3. Complete the remaining patcher options
2) Place the patched dll on the target machine
3) Substituting the patched dll's name and path, silently load the dll using the runll 32 utility. Include the Control_RunDLL flag as shown:

```
>rundll32 C:\elsa.dll,Control_RunDLL
```

4) You should now see rundll32.exe running in the task manager and the elsa dll running using the tasklist /m <dll name> command

5.4 (S) AppInit Mode Installation

(S) In AppInit mode the tool is added to the registry AppInit_DLLs key. Many processes check this key and *automatically load whatever dlls are listed there.* Elsa is configured to verify which process is trying to load it, and only complete loading when the desired process is found. The AppInit mode results in the tool being injected into a legitimate Windows process which can reduce its profile in the task manager.

(S) The Microsoft whitepaper, "AppInit DLLs in Windows 7 and Windows Server 2008 R2" gives a good description of the recent changes to AppInit DLLs. Specifically, these DLLs can require signing with valid certificates. In Windows 7 this is turned off by default and on Windows Server 2008 R2 this is turned on by default. In Windows 2000 SP4, the AppInit technique is not supported. In Windows 8, AppInit is disabled when secure boot is enabled. In these scenarios using an alternate installation mode is recommended.

(S) The 'LoadAppInit_DLLs' value must be set to 1 to turn on AppInit behavior in Windows. The AppInit_DLLs registry value is located in the HKEY_LOCAL_MACHINE hive at the '\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows' subkey. Add it to the AppInit_DLL registry value. If another dll appears at that location use a space or comma as a delimiter to append the short file name for the client DLL to the entry. If the "RequireSignedAppInit_DLLs" is set to 1 then Windows is configured to run signed AppInit DLLs. The DLL must either be signed or this setting must be turned off.

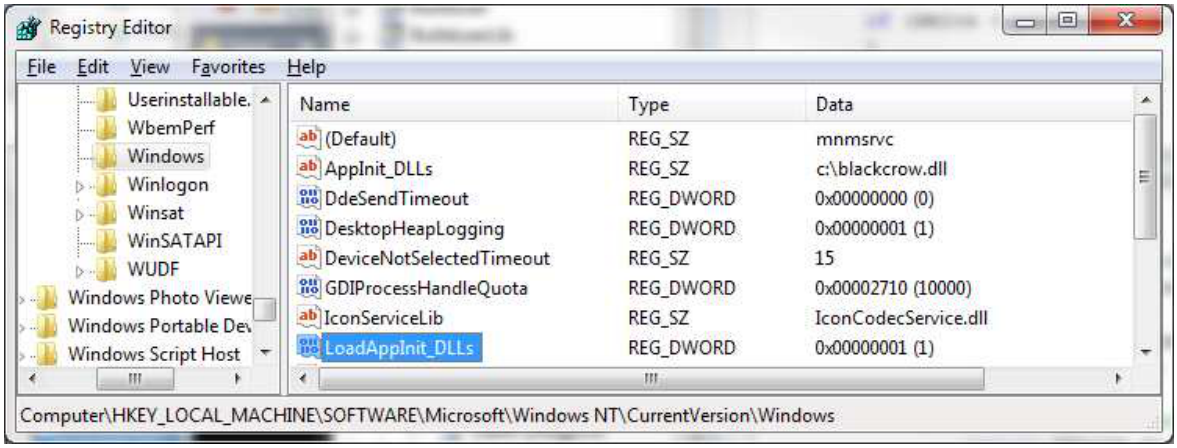

Figure 13 - (S) RegEdit Installation

(S) The REG Windows utility can be used to install the client. As the key is part of HKLM, this operation requires elevated Administrator privileges and should be done at

an elevated prompt. The following commands turn on the Window's AppInit flag and set the path for the client DLL and can be used instead of the regedit GUI:

```
> reg add "HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Windows" /v LoadAppInit_DLLs /t REG_DWORD
/d 1 /f
> reg add "HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Windows" /v AppInit_DLLs /t REG_SZ /d
"C:\elsa.dll" /f
```

Figure 14 - (S) REG Installation Commands

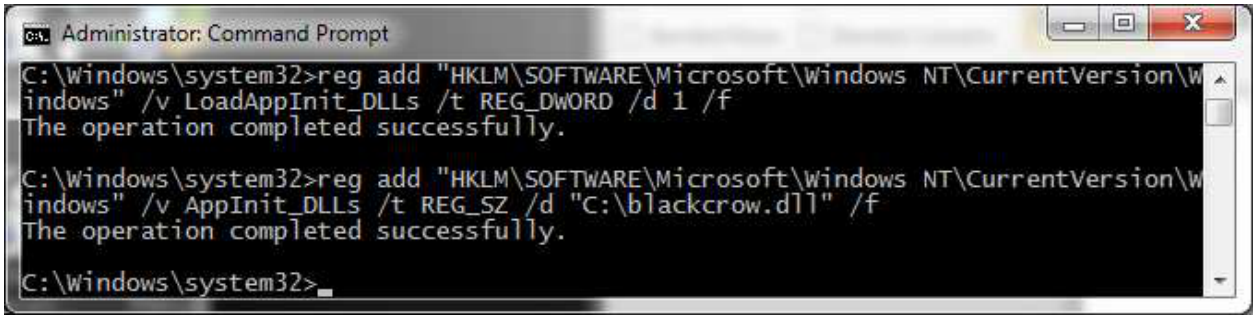

Figure 15 - (S) AppInit Installation Example

(S) In AppInit mode, Windows attempts to load the DLL into every process loaded by USER32. The tool relies on the TargetProcess setting to determine when to begin its execution which also has the effect of controlling the tool's life cycle. When loaded into a process the tool creates a thread and begins its operation. If the process name does not match then the tool is unloaded from that process. If multiple instances of the target process are launched then the first process contains the tool. The tool created a global event using the GUID configuation option to detect previously running instances. This GUID signature can be changed by using the PATCHER tool.

(S) Some Anti-Virus (AV) suites such as Kaspersky and Rising protect critical system process such as SERVICES.EXE and WINLOGON.EXE from AppInit dll injection. These suites can allow injection for non-critical and 3rd party (non-Windows) processes. Deploying ELSA to these systems require careful system survey, targeting, and/ or a cover application for processes vulnerable to this type of injection.

(S) AppInit mode procedures can be useful in installing the tool when the run mode is not AppInit. For example, the operator may elect to have ELSA run as a service dll but install as an AppInit injected dll from inside the 'calc.exe' process. To do this the operator would configure the SvcHost mode and set the InstallFromProcess option as described below to target 'calc.exe'.

6. (S) Persistence and Concealment

(S) The ELSA dll can be renamed without losing its functionality, although certain install methods such as DllHost may require the operator to also rename install scripts to match the ELSA dll. The dll name can also be specified through the PATCHER tool as described above. The dll has no strings in it that would give away its true purpose. The process of patching changes hard-coded bytes in the binary which causes the SHA1 hash to vary slightly. Operator should track file hashes for deployed tools.

(S) The ELSA does not hide it's host process or file. A user watching the process list can see the host process running. This can be mitigated by renaming the dll with an inconspicuous name. Using SvcHost, DllHost, or AppInit installation can reduce the visibility of the program in the process list.

(S) The Elsa dll image and log file reside on disk and hence persist across reboots. However, whether not the Elsa dll runs persistently depends upon how it installed. The SvcHost, DllHost and AppInit installation modes will cause Elsa to run across reboots. The Rundll32 mode will not.

7. (S) Exfiltration

(S) Elsa currently makes no provisions for automatically beaconing or exfiltrating collected data off of the target machine. Therefore the operator must extract the Elsa log file from the target machine through other means.

(S) It is safe (and generally preferable for space savings) to completely remove the Elsa log file; Elsa will detect its absence and create a new one in its place. It is best practice to stop Elsa prior to removing the file, especially if file removal process will be slow. However, simply copying and deleting the file while Elsa runs has not caused problems during testing.

(S) In installations where log files rapidly reach their configured size limit it may be necessary to remove the Elsa file quite regularly to avoid data loss.

8. (U) Processing

(S) Exfiltrated data is symmetrically encrypted and must be processed using the processor tool. This tool has two main functions:

1) Decrypting raw log files generated by Elsa into XML files
2) Resolving lists of wifi access point metadata into geolocations, using the -l <provider> flag

(S) The second step of resolving wifi data into geolocations may have already been performed by Elsa on the target machine if Elsa was configured that way *and* had access to the Internet. Even in that case the operator can re-geolocate saved wifi data from the processor tool, for example to cross reference geolocation information returned from

Microsoft with that Google. See the usage for the Location Provider option below for more detail.

(S) It is important to note that the second step of resolving wifi data into geolocations can also be performed *any* wifi data containing MAC address, SSID and signal strength (for example, netsh output). See the comma separated value option below for more details.

(U) The PROCESSOR command line appears as follows:

```
> processor.exe –i <path> –o <path> [–k <key>] [–f <mask>] [–l provider] [–c]
```

Figure 16 - (S) Command line syntax for the PROCESSOR tool

(U) The subsequent sections describe each option in detail.

8.1 (U) PROCESSOR option Key File

(S) The key file option specifies the path to the encryption key. The switch for this parameter is '-k' and must specify the key file from when the tool was originally configured and patched

8.2 (U) PROCESSOR option Input Directory

(S) The input file or directory option specifies the path to the input file or directory. The switch for this parameter is '-i'.

8.3 (U) PROCESSOR option Output Directory

(S) The output file or directory option specifies the path to the output file or directory. The switch for this parameter is '-o'.

8.4 (U) PROCESSOR option File Wildcard Mask

(S) The file wildcard mask option specifies the file name search filter to use in processing files in the input directory. The default this is '*.*'. The switch for this parameter is '-f'.

8.5 (U) PROCESSOR option Location Provider

(S) The PROCESSOR can be used to relocate wifi access points. This option specifies the location provider to use. The input file option must specify an ELSA output log file or comma separated values containing the wifi access point information. The switch for this parameter is '-l'.

Providers
google
microsoft

Figure 17 - (S) Location provider settings.

8.6 (U) PROCESSOR option Comma Separated Value Format

(S) Elsa A comma separated value file format is also supported. This is useful when access point information can be obtained from other sources such as netsh. The switch for this parameter is '-c', and the file must be formatted as follows:

```
<colon delimited mac address>, <ssid>, <signal strength><carriage return>
```

Figure 18 - (S) Format for csv input.

(S) Note that a script is available for converting data from netsh output into csv files.

```
> netsh wlan show networks mode=bssid
```

Figure 19 - (S) Example command for enumerating wifi networks.

8.7 (U) PROCESSOR example usage

```
 > processor.exe -k key.bin -i indir -o outdir
key    : key.bin
input  : indir
output : outdir
mask   : (null)

processing 'indir\elsa.data' done
processing 'indir\elsa1.data' done
processing 'indir\elsa2.data' done
processing 'indir\elsa3.data' done
processing 'indir\elsag.data' done

5 files processed.
 > processor.exe -i outdir\elsa2.data.xml -o relocate.xml -l microsoft
input    : outdir\elsa2.data.xml
output   : relocate.xml
provider : microsoft
processing Wed Jun 13 14:35:39 2012
processing Wed Jun 13 14:35:43 2012
processing Wed Jun 13 14:36:19 2012
processing Wed Jun 13 14:37:27 2012
```

Figure 20 - (S) Example of command line execution of the PROCESSOR tool

(S) The PROCESSOR generates an output file with the original filename and an .xml file extension. An example entry is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<Log>
    <client>0x1234</client>
    <wifi-ap-list>
        <wifi-ap-entry>
            <timestamp format="UTC">Wed Jun 13 14:35:39 2012</timestamp>
            <flags>0x0</flags>
            <count>11</count>
            <wifi-ap>
                <ssid>linksys</ssid>
                <mac>00:03:52:AB:F4:20</mac>
                <rssi>-37</rssi>
            </wifi-ap>
            <wifi-ap>
                <ssid>Free Public Wifi</ssid>
                <mac>00:04:E2:C6:3B:DA</mac>
                <rssi>-75</rssi>
            </wifi-ap>
            <wifi-ap>
                <ssid>madeup</ssid>
                <mac>00:13:10:C3:26:24</mac>
                <rssi>-75</rssi>
            </wifi-ap>
            <wifi-ap>
                <ssid>TIPICOS GLORIA</ssid>
                <mac>68:7F:74:74:34:2B</mac>
                <rssi>-75</rssi>
            </wifi-ap>
        </wifi-ap-entry>
    </wifi-ap-list>
    <geo-list>
        <geo-entry>
            <timestamp format="UTC">Wed Jun 13 14:35:39 2012</timestamp>
            <accuracy>350</accuracy>
            <provider>microsoft</provider>
            <location>12.3456, -12.34567</location>
        </geo-entry>
    </geo-list>
</Log>
```

Figure 21 - (S) Example of a ELSA xml record

9. (U) Removal

(S) Uninstalling Elsa varies by mode and essentially mirrors the installation process. The following sections describe each mode individually.

9.1 (S) SvcHost Mode Uninstall

(S) To uninstall Elsa in SvcHost mode you must (1) stop the service, (2) unregister it and (3) clean up the Elsa dll and log file.

1) Stop the service, substituting the service name you chose during patching (this can also be done using the services.msc panel):

```
> net stop wmidx
```

2) Silently unregister the service using the RegSvr32 utility:

```
> RegSvr32 /u /s ELSA.DLL
```

3) Remove the dll and log files

9.2 (S) DllHost/Task Scheduler Mode Uninstall

(S) To uninstall Elsa in DllHost mode you must (1) stop the running task, (2) unregister it and (3) clean up the Elsa dll and log file.

1) Stop the task:

```
> schtasks /end /tn "Test Daily Trigger"
```

2) Rename the uninstall script <dll name without '.dll' extension>.uninstall.vbs and place it in the same directory as the Elsa dll
3) Silently unregister the service using the RegSvr32 utility:

```
> RegSvr32 /u /s C:\ELSA.DLL
```

4) Remove the dll and log files. The uninstall script should have disappeared.
5) Verify that the task is no longer in the scheduler:

```
> schtasks /query /tn "Test Daily Trigger"
```

9.3 (S) RunDll32 Mode Uninstall

(S) Simply kill the rundll32 process that loaded Elsa:

1) Find which rundll32 process is running Elsa:

```
> tasklist /m Elsa.dll
```

2) There should only be one entry. Forcefully kill that process using the pid:

```
> taskkill /f /pid <pid from last step>
```

3) Remove the dll and log files

9.4 (S) AppInit Mode Uninstall

(S) To uninstall Elsa running in AppInit Mode:

1) Reset the registry keys edited during install to their original state
2) Remove the log file and dll
3) Elsa may already be running in the process it was injected into. There is currently no method for stopping the dll without stopping that host process. As such the operator must decide whether or not to kill the host process or wait for that process to naturally stop – at which point Elsa will have permanently stopped running.

10. (U) End-to-end Elsa Walk-through

(S) This walk-through describes the configuration, deployment, collection and processing of the ELSA software system and the data it produces. The example is introductory and additional details should be found the prior sections that cover each tool in detail. This section assumes the operator has already stood up all the equipment described in the Prerequisites.

(S) In this example, the operator should setup a Windows machine to serve as the Operator Terminal. The PROCESSOR tool requires Windows so the setup and post processing tools are compatible with Windows 7. In this scenario the operator has extracted the media zip file to a folder 'unclassified\server\windows' on the Operator Terminal as shown below.

```
Directory of C:\Users\user\elsa-v1.0.0-windows\unclassified\server\windows

06/13/2012  10:33 AM    <DIR>          .
06/13/2012  10:33 AM    <DIR>          ..
06/13/2012  09:09 AM             2,717 addtask.vbs
06/13/2012  09:09 AM           116,224 patcher.exe
06/13/2012  09:09 AM           270,336 processor.exe
06/13/2012  09:09 AM               453 sha1-windows-images.txt
06/13/2012  09:09 AM           109,568 tool-x64.dll
06/13/2012  09:09 AM            86,016 tool-x86.dll
               6 File(s)        695,526 bytes
               2 Dir(s)  196,852,690,944 bytes free
```

Figure 22 - (S) Example directory listing of the distribution media

(S) In this example the operator would like to monitor the pattern of life for a target laptop. The operator assigns this collection the operational identifier of 2234, chooses google as the location provider, and will run use rundll32 to host the tool.

```
C:\Users\user\elsa-v1.0.0-windows\unclassified\server\windows
 > patcher.exe -p x64 -o testx64g.dll -W
Enter mode [default is RunDll32]:
Enter target process filename [default is rundll32.exe]:
Enter data file name [default is %SystemRoot%\TEMP\elsa.data]:
%SystemRoot%\TEMP\elsag.data
Enter data file max kb [default is 200]: 202
Enter data file archive seconds [default is 60]: 62
Enter data encryption key file [default is key.bin]:
Create a new application guid [default is no]:
Enter application guid [default is {59553112-3228-49ce-8044-4AB3C63BD46C}]:
Enter seconds between wifi surveys [default is 30]: 32
Enter the seconds to delay the wifi survey after install [default is 30]: 32
Enter the seconds to delay the wifi survey after startup [default is 30]: 32
Enter backoff factor to use when wifi survey are unsuccessful [default is 10]: 13
Enter the wifi rssi threshold [default is 100]: 202
Save all wifi surveys [default is no]:
Enter geolocation provider [default is google]:
Enter the Client Id [default is 5555]: 2234

C:\Users\user\elsa-v1.0.0-windows\unclassified\server\windows
 > dir
```

```
 Volume in drive C is OS
 Volume Serial Number is 16CB-523E

 Directory of C:\Users\user\elsa-v1.0.0-windows\unclassified\server\windows

06/13/2012  10:44 AM    <DIR>          .
06/13/2012  10:44 AM    <DIR>          ..
06/13/2012  09:09 AM             2,717 addtask.vbs
06/13/2012  10:33 AM               130 key.bin
06/13/2012  09:09 AM           116,224 patcher.exe
06/13/2012  09:09 AM           270,336 processor.exe
06/13/2012  09:09 AM               453 sha1-windows-images.txt
06/13/2012  10:44 AM           109,568 testx64g.dll
06/13/2012  09:09 AM           109,568 tool-x64.dll
06/13/2012  09:09 AM            86,016 tool-x86.dll
06/13/2012  10:44 AM               512 wizard.config
              8 File(s)        806,919 bytes
              2 Dir(s)  196,851,429,376 bytes free
```

Figure 23 - (S) Example PATCHER session

(S) On the target system proceed with the execution of the tool using the following command. Note the command line is case sensitive for the DLL export string 'Control_RunDLL'.

```
 > rundll32 testx64g.dll,Control_RunDLL
```

Figure 24 - (S) Example of start of ELSA collection module

(S) ELSA will begin monitoring wifi networks. It will wait until the first archive period before writing the log file. In this example the archive will be update every 62 seconds. The tool will wait for the startup and install delays to expire before collecting wifi observations. As configured this will be 32 seconds. The operator can verify the tool is running by observing the '%SystemRoot%\TEMP\elsag.data' for file size changes. Wireshark is useful for observing the HTTPS traffic as the tool queries google or microsoft for geolocation coordinates. After several wifi survey intervals the operator can connect and copy the collection file '%SystemRoot%\TEMP\elsag.data' into a new directory 'outdir' on the Operator Terminal and process it.

```
C:\Users\user\elsa-v1.0.0-windows\unclassified\server\windows
 > mkdir indir
C:\Users\user\elsa-v1.0.0-windows\unclassified\server\windows
 > mkdir outdir
C:\Users\user\elsa-v1.0.0-windows\unclassified\server\windows
 > processor.exe -k key.bin -i indir -o outdir
key    : key.bin
input  : indir
output : outdir
mask   : (null)

processing 'indir\elsag.data' done

1 files processed.

C:\Users\user\elsa-v1.0.0-windows\unclassified\server\windows
```

```
> notepad outdir\elsag.data
```

Figure 25 - (S) Example of decrypting an ELSA encrypted archive

(S) The operator uses notepad to view the contents of the output xml file.

```
<?xml version="1.0" encoding="UTF-8"?>
<Log>
    <client>0x2234</client>
    <wifi-ap-list>
        <wifi-ap-entry>
            <timestamp format="UTC">Wed Jun 13 14:42:27 2012</timestamp>
            <flags>0x0</flags>
            <count>12</count>
            <wifi-ap>
                <ssid>BREAD SHOP</ssid>
                <mac>00:03:52:AB:F4:20</mac>
                <rssi>-29</rssi>
            </wifi-ap>
            <wifi-ap>
                <ssid>CableCo</ssid>
                <mac>00:04:E2:C6:3B:DA</mac>
                <rssi>-77</rssi>
            </wifi-ap>
            <wifi-ap>
                <ssid>grace</ssid>
                <mac>00:13:10:C3:26:24</mac>
                <rssi>-73</rssi>
            </wifi-ap>
            <wifi-ap>
                <ssid></ssid>
                <mac>00:14:D1:AE:D3:EC</mac>
                <rssi>-81</rssi>
            </wifi-ap>
            <wifi-ap>
                <ssid></ssid>
                <mac>00:14:D1:AE:D3:FC</mac>
                <rssi>-81</rssi>
            </wifi-ap>
            <wifi-ap>
                <ssid>09BX03031164</ssid>
                <mac>00:23:97:33:5D:F1</mac>
                <rssi>-79</rssi>
            </wifi-ap>
            <wifi-ap>
                <ssid>09FY11030962</ssid>
                <mac>00:23:97:C0:9E:11</mac>
                <rssi>-81</rssi>
            </wifi-ap>
            <wifi-ap>
                <ssid>dlink</ssid>
                <mac>1C:AF:F7:DB:0B:B7</mac>
                <rssi>-83</rssi>
            </wifi-ap>
            <wifi-ap>
                <ssid>Belkin46</ssid>
                <mac>68:7F:74:74:34:2B</mac>
                <rssi>-77</rssi>
            </wifi-ap>
            <wifi-ap>
                <ssid></ssid>
                <mac>BC:AE:C5:C7:A6:77</mac>
                <rssi>-79</rssi>
```

```
            </wifi-ap>
            <wifi-ap>
                <ssid>Cisco12345</ssid>
                <mac>C0:C1:C0:F5:B2:16</mac>
                <rssi>-81</rssi>
            </wifi-ap>
            <wifi-ap>
                <ssid>Madison113</ssid>
                <mac>C4:3D:C7:AD:96:72</mac>
                <rssi>-81</rssi>
            </wifi-ap>
        </wifi-ap-entry>
    </wifi-ap-list>
    <geo-list>
        <geo-entry>
            <timestamp format="UTC">Wed Jun 13 14:39:25 2012</timestamp>
            <accuracy>350</accuracy>
            <provider>google</provider>
            <location>38.12345, -77.12345</location>
        </geo-entry>
        <geo-entry>
            <timestamp format="UTC">Wed Jun 13 14:35:39 2012</timestamp>
            <accuracy>350</accuracy>
            <provider>google</provider>
            <location>38.12345, -77.12345</location>
        </geo-entry>
        <geo-entry>
            <timestamp format="UTC">Wed Jun 13 14:35:43 2012</timestamp>
            <accuracy>350</accuracy>
            <provider>google</provider>
            <location>38.12345, -77.12345</location>
        </geo-entry>
        <geo-entry>
            <timestamp format="UTC">Wed Jun 13 14:40:25 2012</timestamp>
            <accuracy>350</accuracy>
            <provider>google</provider>
            <location>38.12345, -77.12345</location>
        </geo-entry>
        <geo-entry>
            <timestamp format="UTC">Wed Jun 13 14:36:19 2012</timestamp>
            <accuracy>350</accuracy>
            <provider>google</provider>
            <location>38.12345, -77.12345</location>
        </geo-entry>
        <geo-entry>
            <timestamp format="UTC">Wed Jun 13 14:41:25 2012</timestamp>
            <accuracy>350</accuracy>
            <provider>google</provider>
            <location>38.12345, -77.12345</location>
        </geo-entry>
        <geo-entry>
            <timestamp format="UTC">Wed Jun 13 14:38:25 2012</timestamp>
            <accuracy>350</accuracy>
            <provider>google</provider>
            <location>38.12345, -77.12345</location>
        </geo-entry>
    </geo-list>
</Log>
```

Figure 26 - (S) Example of an ELSA xml collection

(S) This log file contains several coordinate pairs for the target machine as well as a single wifi ap list entry. Because we selected the default option to delete geolocated wifi

access point observations this wifi entry must not have been geolocated. This could be a recent entry collected when the laptop could see wifi access points even though it was not actually connected to the Internet over any of them. Hence ELSA was able to observe nearby wifi networks but not able to resolve those networks to a geo. The operator can now use the processor try to resolve that wifi data to a geolocation from the Operator Terminal at base.

```
C:\Users\user\elsa-v1.0.0-windows\unclassified\server\windows
 > processor.exe -i outdir\elsag.data.xml -o relocateg.xml -l google
input    : outdir\ elsag.data.xml
output   : relocateg.xml
provider : google
processing Wed Jun 13 14:42:27 2012
```

Figure 27 - (S) Example of the PROCESSOR relocating a wifi access point

(S) Viewing the relocateg.xml log in notepad shows the successful result of the query.

```
<?xml version="1.0" encoding="UTF-8"?>
<Log>
    <geo-list>
        <geo-entry>
            <timestamp format="UTC"> Wed Jun 13 14:42:27 2012</timestamp>
            <accuracy>75</accuracy>
            <provider>google</provider>
            <location>38.123456789, -77.123456789</location>
        </geo-entry>
    </geo-list>
</Log>
```

11. (U) Additional Operational Procedures

11.1 (S) InstallFromProcess Mode

(S) The InstallFromProcess configuration option can be used to configure the DLL to install from a targeted process. With this option set when the DLL is injected into a process it can be configured to run its installation routines to complete its setup. In most cases the process would need to run in an elevated Administrator or SYSTEM context.

(S) The DLL looks for a trigger file as part of this action. After the DLL runs through the install or uninstall procedures it deletes the trigger file to prevent repeat install operations. The trigger file is a zero length file with the same name as the dll using a different file extension as specified below.

Extension	Notes
.install	When the DLL is injected into a process matching the InstallFromProcess and the zero length file is found in the same directory then the DLL will run through its installation routines. If the dll is named 'foo.dll' then the zero length file should be named 'foo.install' and reside in the same directory.
.uninstall	When the DLL is injected into a process matching the InstallFromProcess and the zero length file is found in the same directory then the DLL will run through its installation routines. If the dll is named 'foo.dll' then the zero length file should be named 'foo.uninstall' and reside in the same directory.

Figure 28 - (S) File extension information for Dll Injection.

12. (U) Troubleshooting and Support

12.1 (U) Assistance and Problem Reporting

(S) The ELSA software system is maintained by IOC/EDG. Assistance and problem solving is provided by IOC/EDG/AED/RDB and problem reporting is provided by IOC/COG/OED/GB. Please see the Troubleshooting section of this document for more information.

(S) The IOC Collaboration Portal, Bulletin Board System (BBS) exists for question and answer. The BBS does not provide mechanism notifying us when topics are created but its use provides a common forum for lessons learned. Topics should be created and prefixed using the tool name and version e.g. ' ELSA v3.0 ...' When in doubt the tool version is included in the sha1 hash files from the installation media.

12.2 (U) Troubleshooting Guide

#	Symptom	Resolution
1	No log file appears	▲ Check to see **if the dll is loaded** using `tasklist /m <elsa dll name>`. If not, see the appropriate 'Dll not loaded' entry below ▲ Check to see **if the host process has write permission** in the selected log directory. For example, rundll32 will not have write permission in system directories if run from a non-elevated prompt
2	Dll not loaded – AppInit Mode	▲ Check to see **if the selected host process has restarted** since the registry keys were edited ▲ Check to see **if the selected host process loads user32.dll** using `tasklist /m user32.dll` while the host process is running. If not you must select a different host process ▲ Check to see **if the target OS supports AppInit mode** using the install section above. ▲ Check to see **if the registry keys were configured according to the install section above**
3	Dll not loaded – DllHost Mode	▲ Check to see **if the elsa task appears in the task scheduler** using the active tasks panel in the taskschd.msc UI. If not, modify the install script until it successfully adds the task ▲ Check to see **if the elsa task has run yet** by double clicking on the task in the taskschd.msc

		UI and looking at the 'Last Run Time' field. If it has run yet, either wait for it to run or re-install with a different run time • Check to see **if the elsa task completed successfully** by double clicking on the task in the taskschd.msc UI and looking at the 'Last Run Result' field. If not, check that the Class ID in the install script matches the GUID specified in the patcher
4	Dll not loaded – SvcHost Mode	• Check to see **if the elsa service appears in the services.msc UI or task manager services tab**. If not, double check the install against the svchost install procedure above • Check to see **if the elsa service is listed as started in the services.msc UI or task manager services tab**. If not, manually start it using `net start <service name>`
5	Dll not loaded – RunDll Mode	• Double check the install against the svchost install procedure above, particularly the Control_RunDLL flag
6	No geo results in decrypted log file	• Check to see **if wifi access points are present**. If they are, use the processor to geolocate the results • Check to see **if elsa was configured with a geolocation provider** results. If not, re-patch and reinstall
7	Fewer wifi results than expected in decrypted log file	• Check to see that **the log file size was not close to the configured limit** • Check to see that **if the wifi rssi threshold was high (e.g. - 60 dBm or more)**. Re-patch and reinstall with a very low (e.g. - 5 dBm) setting.
8	Alphanumeric (base64 encoded) values in geolocation results	• Check to see **if the alphanumeric values match any of the specific cases below**. • This indicates a parser error, see 'interpreting errors in Elsa xml files' below • It is recommended that you configure Elsa to retain wifi ap lists for geolocation using the processor if these errors are common
9	'H4sI' in microsoft location results	• This appears intermittently in microsoft results and is believed to be a random server error. Attempt to re-geo the result.
10	'b3VyY2U9 …' in microsoft location results	• This decodes to an alternate geolocation format returned by microsoft. Although this is under investigation it does not appear that a lat/lon is

		present in these results. Hence it appears to indicate that Microsoft's database does not contain the provided wifi data.
11	'TFRTIgp9Cg==' in google location results	⅄ This decodes to 'LTS"\n}\n', which is the end of google's 'NO RESULTS' message. This occurs when google has no results in its database. Future versions of Elsa will communicate this more clearly in the results
12	Valid lat/lons accompanied by high (poor) accuracy values	⅄ Check to see **if the latitude and longitude fields contain valid results**. If not, see 'Alphanumeric (base64 encoded) values in geolocation results ⅄ If a lat/lon is present then Elsa successfully queried the provider and t**his indicates that their database does not contain an accurate geo for the given AP lists**. Sometimes Microsoft will fall back to basic (inaccurate), ip-based geolocation in these cases.
13	Log file keeps appearing after uninstall	⅄ Check to see **which process is loading the dll** using `tasklist /m <elsa dll name>`. Use the uninstall procedures above to prevent that process from loading the dll again, then restart that process.
14	net start or sc start commands fail with error 1060	⅄ Check to see that the **platform architecture matches the patcher –p option** using: `wmic cpu get description` ⅄ Check that the platform architecture of **the host process matches the target computer** e.g. 32 bit Taper on x64 Windows. If this is the case then install using: `%windir%\sysnative\regsvr32.exe /s %windir%\elsax64.dll` ⅄ Move the dll to a directory other than `C:\Windows\system32`

15	wlansvc or eaphost services not running	⅄ Configure the services as follows:
		`sc config eaphost start= demand`
		`sc config wlansvc start= demand`

12.3 (U) Interpreting Errors in ELSA xml files

(S) If Elsa is able to transmit a wifi geolocation query, it will attempt to parse whatever response it receives into a geolocation. If it is unable to find a geolocation in the response it will base 64 encode the region of the file where the geolocation usually turns up and place that in the geolocation field for offline analysis. You can use python's b64decode functionality to inspect the contents of these fields.

(S) It is important to note that Elsa currently will not attempt to re-resolve a parse failure. It is assumed that subsequent resolution attempts will produce similar errors, which could be alerting to the server. However, there are certain cases where this is not true, especially in the presence of the 'captive portals' or 'paywalls' often encountered at hotels and Internet cafes. In these situations, the tool may detect the presence of an active Internet connection before the user has clicked through the paywall. In that case the tool will send geolocation requests before a connection is truly available to the 3rd party database. Since these types of access points resolve ALL url requests to the paywall webserver Elsa will unsuccessfully search through the returned paywall HTML for the geolocation.

(S) The upshot of this is that the most reliable way to obtain geolocations is to save all wifi access point information so that it can be used to resolve locations offline. If the current version of the tool is set to delete access point information after a geolocation attempt, the ability to obtain any geolocation after the fact may be lost.

13. (U) IV&V Findings and Observations from testing

(U) This section is intentionally blank until testing is completed.

14. (U) Scripts for DllHost Mode Install / Uninstall

```
' This sample schedules a task to start on a daily basis.

' A constant that specifies a daily trigger.
const TriggerTypeDaily = 2
' A constant that specifies an executable action.
const ActionTypeExec = 0
const ActionTypeCom = 5

' Create the TaskService object.
Set service = CreateObject("Schedule.Service")
call service.Connect()

' Get a folder to create a task definition in.
Dim rootFolder
Set rootFolder = service.GetFolder("\")

' The taskDefinition variable is the TaskDefinition object.
Dim taskDefinition
' The flags parameter is 0 because it is not supported.
Set taskDefinition = service.NewTask(0)

' Set the task setting info for the Task Scheduler by creating a TaskSettings object.
Dim settings
Set settings = taskDefinition.Settings
settings.Enabled = True
settings.StartWhenAvailable = True
settings.Hidden = False

' Create a daily trigger. Note that the start boundary  specifies the time of day
' that the task starts and the interval specifies what days the task is run.
Dim triggers
Set triggers = taskDefinition.Triggers

Dim trigger
Set trigger = triggers.Create(TriggerTypeDaily)

' Trigger variables that define when the trigger is active  and the time of day
' that the task is run. The format of this time is YYYY-MM-DDTHH:MM:SS
Dim startTime, endTime
Dim time
startTime = "2011-10-19T10:30:00"
endTime = "2012-05-02T10:30:00"

WScript.Echo "startTime :" & startTime
WScript.Echo "endTime :" & endTime

trigger.StartBoundary = startTime
trigger.EndBoundary = endTime
' Task runs every day.
trigger.DaysInterval = 1
trigger.Id = "DailyTriggerId"
trigger.Enabled = True

' Set the task repetition pattern for the task. This will repeat the task 5 times.
Dim repetitionPattern
Set repetitionPattern = trigger.Repetition
repetitionPattern.Duration = "PT4M"
repetitionPattern.Interval = "PT1M"

' Define information about the task. Set the registration info for the task by
' creating the RegistrationInfo object.
Dim regInfo
Set regInfo = taskDefinition.RegistrationInfo
regInfo.Description = "Test Task"
regInfo.Author = "Administrator"
```

```
' Create the action for the task to execute.

' Add an action to the task to run
Dim Action
Set Action = taskDefinition.Actions.Create( ActionTypeCom )
' Set the class id to the guid
Action.ClassId = "{9A01ADFB-8B6F-4E19-8334-1A17C6FB7402}"

WScript.Echo "Task definition created. About to submit the task..."

' Register (create) the task.
call rootFolder.RegisterTaskDefinition( "Test Daily Trigger", taskDefinition, 6, , , 3)

WScript.Echo "Task submitted."

' This will delete the task named "Test Daily Trigger"
'Set objShell = WScript.CreateObject("WScript.Shell")
'objShell.Run ("schtasks /delete /f /tn ""Test Daily Trigger"""), 0, false
```

Figure 29 - (S) Task Scheduler Install Script

```
' This will delete the task named "Test Daily Trigger"
Set objShell = WScript.CreateObject("WScript.Shell")
objShell.Run ("schtasks /delete /f /tn ""Test Daily Trigger"""), 0, false
```

Figure 30 - (S) Task Scheduler Uninstall Script