cyber RangeKit

Tools to Validate Your Security Implementation

Greg Hoglund

VP Enterprise Security, Symantec Corporation

Purpose of the cyber RangeKit

- Create an independent toolkit for testing EDR/EPP efficacy
  - There is a general lack of robust or effective QA and testing in the industry
- Provide a way to validate that EDR/EPP implementation is working at the endpoint in production
  - Even good solutions can have implementation problems or have gaps
- Intention to be used both in the lab and production environments
  - Use in production drove most of the tool architecture
- Exercise event pathways and SoC workflows

A healthy tension

- Open-source and extensible by the user community
  - Easy to modify so users can make variations on theme
  - Thus, not a static target for vendor QA
- Put the power in the hands of the users and customers, instead of relying solely on the claims of vendors
- Give vendors a good QA toolkit
  - For example, by focusing on reproducibility
- This is a healthy tension to push the defensive envelope forward

What People Have Asked For?

- Go beyond faith - validate vendor claims (with relative low cost)
- Ensure security technology is not ‘rigged’ with static/simplistic signatures
  - Real attackers don’t use Metasploit – this isn’t a lab
- Measure detection/prevention efficacy (the ‘risk floor’ - nothing is 100%)
  - Pre-req to setting business goals to increase coverage
- Ensure that implementation of technology in-field meets minimum functional requirements of detection, prevention, and performance
  - Beyond just ICAR test virus which only shows it’s plugged in
- Make actionable tests from well socialized and understood attack patterns (like MITRE ATT&CK knowledge base)
  - There is high demand to build risk prevention frameworks and strategies around this

Augment or Alternative to Red Teams

- NOT for vulnerability assessment, but testing defenses
- Red Teams can provide much of these needs, BUT
  - Limited to large organizations with large security teams and funding
  - Requires hard to find skill set – engineers w/ security skill == talent shortage
  - Managing a red team also requires org to have high security maturity
- Red teams tend to rely on human labor
  - Testing is not always consistent / aka ‘following rabbit holes’
  - Often, artifacts are left behind that then result in false positives
  - Frequency of security validation may be too slow
  - Expensive $$$$

Prior Work

- https://github.com/api0cradle/LOLBAS
- https://github.com/endgameinc/RTA
- https://github.com/redcanaryco/atomic-red-team
- https://github.com/uber-common/metta
- https://github.com/mitre/caldera

Please let me know who I missed…

Commercial Efforts

- Gartner → "Breach and Attack Simulation"
- AttackIQ
- SafeBreach
- Verodin
- vThreat/ThreatCare
- NSS Labs

Please let me know who I missed…

Cyber Validation – Potential Features

- Integration to SoC where toolset will validate security products
  - Detected an event
  - Extracted the appropriate evidence for the event
  - For timeline scenarios, which steps of attack were detected or blocked
  - Which controls are working, which ones need work?
- Evaluate
  - Was the event actionable?
  - How did security personnel respond?
  - Is the orchestration playbook effective?

Cyber Validation – Potential Features

- Perform validation measurement over time
  - What security controls have degraded over time?
  - Did a gap suddenly emerge?
  - Is there overlap between two or more controls? Can one be retired?
  - Detect what is improving
- Demonstrate to executive management the ongoing value of cyber dollars – without having to suffer a breach to prove it

Some Terminology

Symantec™

Some Terminology (my version anyway 😐)

- Observable - Objects that can be created, modified, or otherwise controlled that can also be measured by security tools / products
  - Including associated metadata, timestamp, filesize, etc.
- Artifact – a registry key, task, or other observable that serves as evidence of an attack. By itself not a threat, but data indicator.
- Attack Pattern – a set of artifacts that, when seen together, indicate a potential compromise or threat activity
  - Example, attack patterns the correlate with MITRE Att&ck framework
- Decoy – an INERT/SAFE file or object that, if actually real, would be a threat. Decoy’s are an absolute requirement so there is no instability or threat to the environment.

Symantec.

More Terminology

- Breach Scenario – a defined attack pattern that probably consists of multiple artifacts. The testing will exercise variations of the breach scenario. This is what you talk about to upper management.
- Continuous Security Validation – a company strategy that assumes security systems will begin to fail over time and need consistent attention to stay current and functional within the enterprise

What is an Artifact?

- Data structure left in storage on an endpoint
  - Storage is in-memory, or on-disk
- All artifacts are created by software
  - More specifically, a <u>running</u> software process
- A Software process, by proxy, represents
  - Direct Human User-action
  - Network activity / transaction (packets coming and going)
  - Automated process or loop (pre-programmed)

What is an Attack Pattern?

- Combinations of artifacts together to make a pattern (splash pattern)
- The combination can be correlated to a specific event
- Types of patterns
  - Software Exploitation
  - Malware Installation and Deployment
  - Command and Control
  - Interaction with the Host
  - Lateral Access

Architecture Features

to support use in production and as QA tool

Symantec™

Summary of Architecture

- Simulates attack patterns by creating forensic observables
  - Some low level code is required, some file reverse engineering
  - Or, alternatively, executes behavior / API calls but without causing harm
- Uses DECOY payloads so no actual threat activity takes place
- Ability to clean-up artifacts after test (aka no-trace)
- Has test reproducibility

Forensic Simulation

- Tool simply creates appropriate files, keys, and other observables.
- In some cases only filename has to be correct – perhaps in other cases the contents of the file must also be simulated
  - This depends on how stringent the EDR solution is with regard to examination
- Timestamp control is very important
  - TIME is a key component to creating a cohesive attack pattern
- Some artifacts can be <u>induced</u> rather than directly created
  - You can create a fake crashdump file. But, you can also inject a thread into the browser and have that thread crash the process. The latter will induce the creation of a crashdump.

Symantec.

A note on Behavioral Simulation

- Simulating forensic artifacts helps we post-compromise assessment – tools that actually examine artifacts-at-rest
- However, with recording agents, the tools have to preform actual API calls – this also creates artifacts, but more importantly is simulates actions
  - Recording agents rely almost entirely on API hooks and kernel-mode filter drivers to capture API-calls in near-realtime
- Both approaches need to be addressed for the RangeKit to be complete

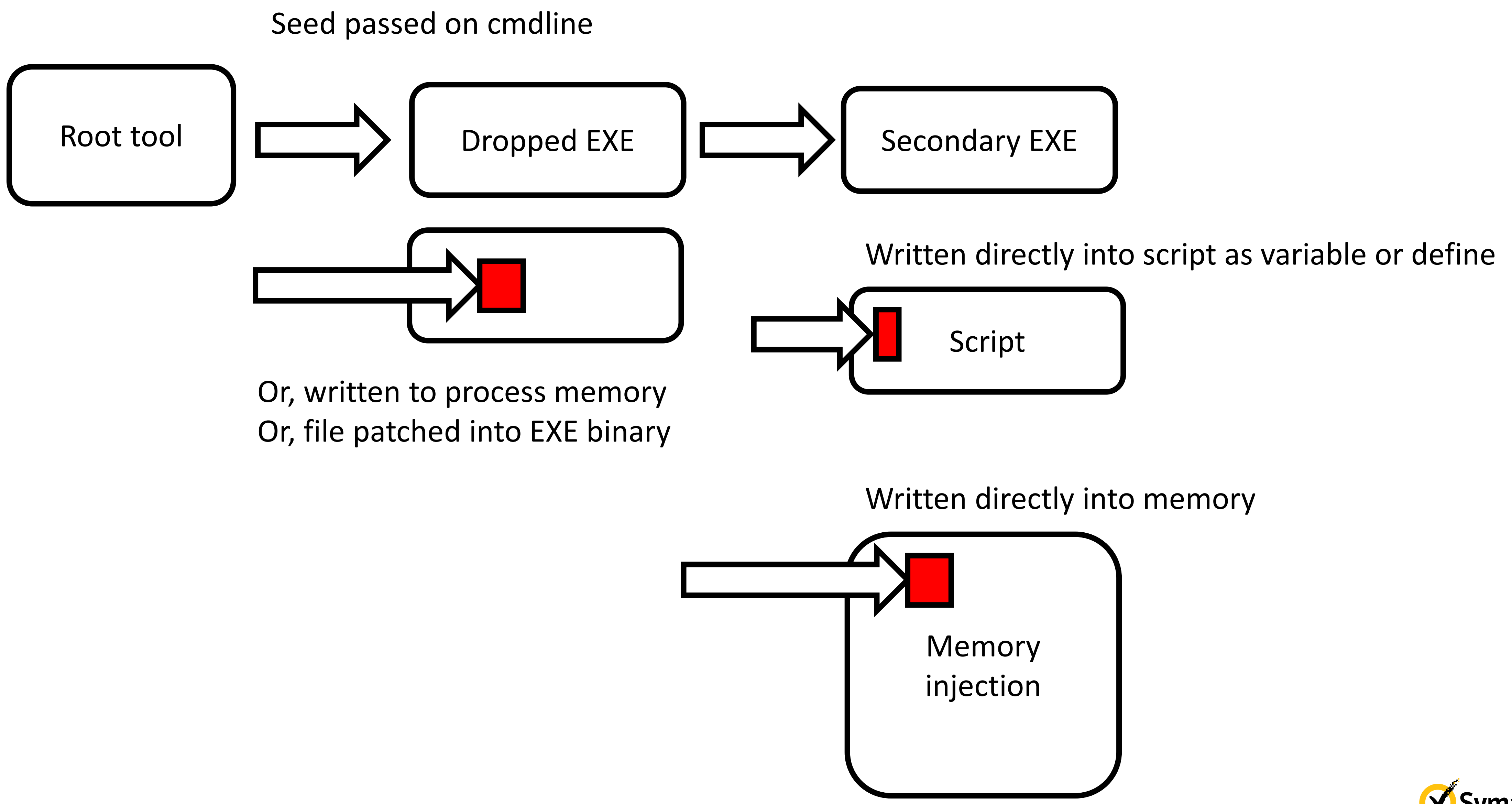
Concept: Decoy Cohesion

- Since decoys aren’t real how do you ensure this is a valid test?
- Has enough ‘cohesion’ to trigger EDR/EPP products
  - Pattern has enough behavioral or signature traits to trigger EPP/EDR
- There could be debate here – some purists may argue that actual threat behavior must execute or the test is not valid
  - I argue there is a threshold of ‘cohesion’ where enough activity is simulated but without taking any final steps that would exercise a true threat situation
- No damage to assets. No true data exfiltration. No backdoors. No credential compromise.
  - Basically user doesn’t have to ‘fix’ anything post-test. For example, they don’t have to change any account passwords.

No-trace and Downstream Positive Control

- The root tool has the ability to generate names/values of all downstream artifacts
- Can create digest of these values without actually running test
- Because these names are known pre-test, the root tool can clean the artifacts post-test
- Child observables don't have to perform cleanup
  - Makes them more realistic, don't have to stay resident
  - Example, imagine a self-deleting EXE can't stay resident
- No central audit log that can easily be used as a crutch for detection
  - EDR/EPP can't just look for handles to the audit log and find the artifacts

All child components can generate ‘temp’ values, filenames, keys, or other observables using derivatives of seed value.


Seed passed on cmdline
Root tool
Dropped EXE
Secondary EXE
Written directly into script as variable or define
Script
Or, written to process memory
Or, file patched into EXE binary
Written directly into memory
Memory injection

Symantec.

DecoyName.exe <seed value> RUN|DEPLOY

```
if (3 == argc)
{
        int seed_value = _wtoi(argv[1]);
        if (0 == seed_value && EINVAL == errno)
        {
                wprintf(L"! When setting the seed you must use a valid integer value.\n");
                exit(EXIT_FAILURE);
        }

        if (!wcscmp(argv[2], L"RUN"))
        {
                while (true)
                {
                        DoSomething(seed_value);
                        Sleep(1000);
                }
        }
        else
        if (!wcscmp(argv[2], L"DEPLOY"))
        {
                MakeCopyAndExecute(seed_value);
        }
```

Extremely simplistic generator using seed

```
BOOL GenerateFileName(DWORD key, WCHAR *outFileName)
{
        _snwprintf_s(outFileName, MAX_PATH - 2, _TRUNCATE, L"%ws%08X.bin", L"testfile", key);
        return TRUE;
}
```

Obviously generator algorithms can be complex. It's a matter of choice. It's also one of the easiest components to modify in source-code. Users are encouraged to create their own variants so pre-canned name signatures won't work on the RangeKit tool.

Test Reproducibility

- This tool is meant to help security vendors make better products
  - And to help users fix implementation problems, of course
- As a QA tool, reproducibility is King
- This is why Downstream-positive-control system is seed-able
- Use the same options and same seed, in theory you have same test
  - Different OS environments may have different host variables, environment variables, users, etc… so some artifacts may differ with regard to file path. However, the behavior is almost exactly the same and [in many cases] close enough for reproducibility.
- Easy to communicate test condition from customer to vendor
  - A notoriously hard problem in most cases, made easy (easier, anyway)

Symantec.

How to Simulate Attacker Activity

Technically Like Deception, but for Validation

- The technology is technically similar to deception
- However, the use case is NOT for catching real attackers
- NOT used as honeypot
- Intent is to trigger / exercise security frameworks and response processes in the SoC <u>without actually suffering a real attack</u>
  - No live munitions, but still be cohesive

```
HKEY hKey = NULL;
HKEY hKeyRoot = HKEY_CURRENT_USER; // of HKEY_CLASSES_ROOT HKEY_CURRENT_CONFIG HKEY_CURRENT_USER HKEY_LOCAL_MACHINE HKEY_USERS

status = RegOpenKeyEx(
                        hKeyRoot,
                        L"Software\\Microsoft\\Windows\\CurrentVersion\\Run",   // Note key names are not case sensitive.
                        0,
                        KEY_SET_VALUE,
                        &hKey);

if (status != ERROR_SUCCESS)
{
        xprintf(OPLOG_NORMAL, L"! Could not open Run key. Error:%08X\n", GetLastError());
        return FALSE;
}

WCHAR *some_data = theFilePath;
WCHAR *some_value = theValueName;

status = RegSetValueEx(
                        hKey,
                        some_value,
                        0,                                                                              // reserved
                        REG_SZ,                                                                 // type of value
                        (const BYTE *)some_data,
                        ((DWORD)wcslen(some_data) + 1) * 2);    // length IN BYTES including NULL CHARACTER!
```

Creating a value in the Run key.

This doesn't actually run the EXE – it doesn't have to. The presence of the key/value and making it point to an EXE on disk is enough to test EDR/EPP.

Why not a driver?

- A filter driver or hooking driver could also create simulated observables in-situ
- BUT, it won’t create user-mode side API activity – so behavior monitoring may be more difficult to exercise
- And, multiple hooking/filtering drivers (due to EDR use of the same) could create some interoperability issues
- Memory injection, etc., is possible to simulate from kernel
- I am open to suggestions here, but for now I have been focused on creating/destroying true artifacts

Three primary relationships

- Relationships via TIME
  - Ensure that the artifacts fit into a timeline
- Relationships via USER
  - Ensure the artifacts are owned by the proper SID, have correct ACL’s
- Relationships via SOFTWARE
  - Research the software and it will show you the artifacts that are part of the attack pattern

Creating Patterns: Intrinsic Relationships

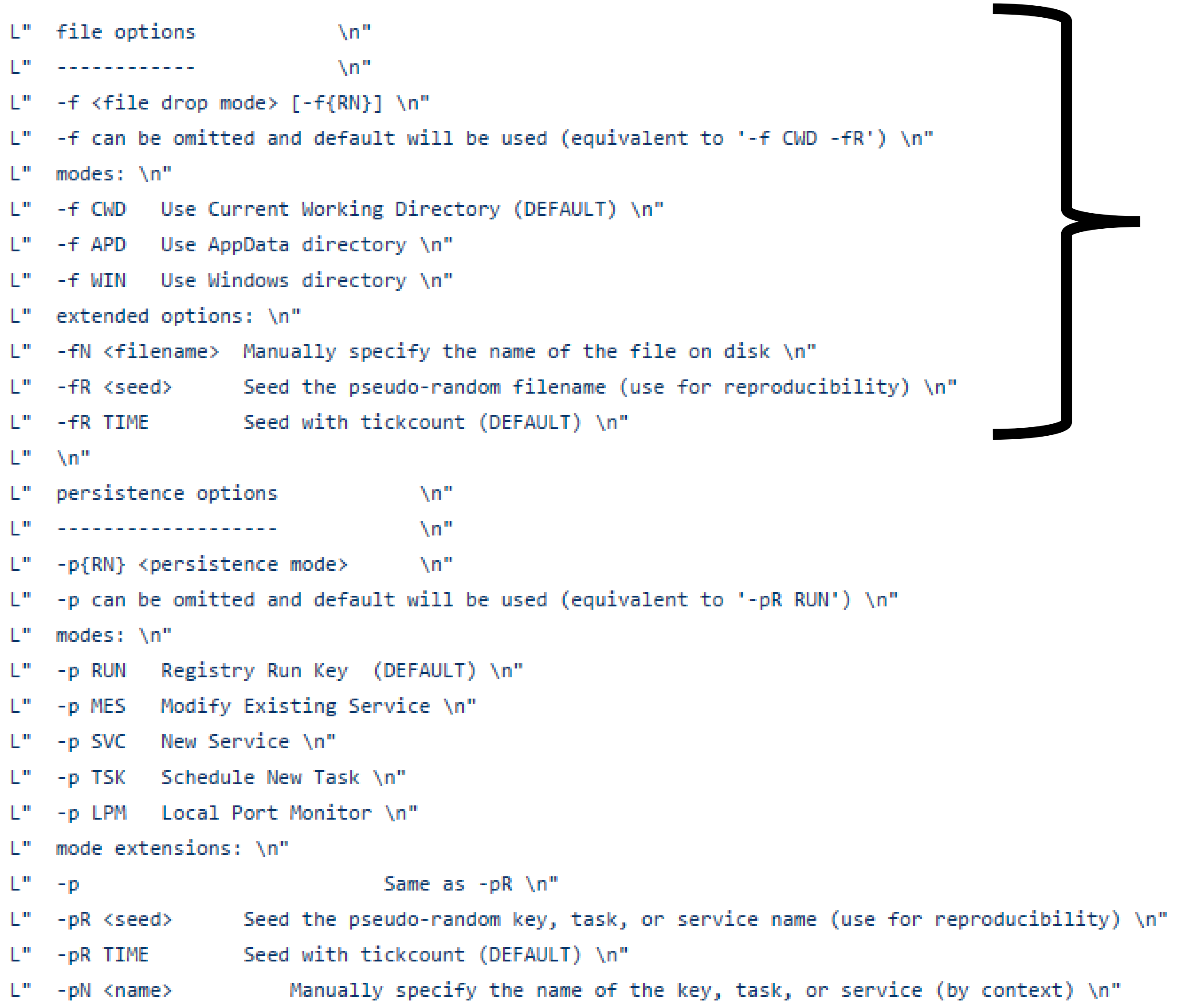
Artifacts have built-in relationships, for example

- Process->Modules->Files
- Process->Network
- Regkeys->Files
- User->Process

These relationships can be used to create patterns.

The RegKey->File relationship.

Imagine the EDR produce sees one of these startup keys pointing to an EXE in one of these directories… Suspicious?



```
L"  file options            \n"
L"  ------------            \n"
L"  -f <file drop mode> [-f{RN}] \n"
L"  -f can be omitted and default will be used (equivalent to '-f CWD -fR') \n"
L"  modes: \n"
L"  -f CWD   Use Current Working Directory (DEFAULT) \n"
L"  -f APD   Use AppData directory \n"
L"  -f WIN   Use Windows directory \n"
L"  extended options: \n"
L"  -fN <filename>  Manually specify the name of the file on disk \n"
L"  -fR <seed>      Seed the pseudo-random filename (use for reproducibility) \n"
L"  -fR TIME        Seed with tickcount (DEFAULT) \n"
L"  \n"
L"  persistence options            \n"
L"  -------------------            \n"
L"  -p{RN} <persistence mode>      \n"
L"  -p can be omitted and default will be used (equivalent to '-pR RUN') \n"
L"  modes: \n"
L"  -p RUN   Registry Run Key  (DEFAULT) \n"
L"  -p MES   Modify Existing Service \n"
L"  -p SVC   New Service \n"
L"  -p TSK   Schedule New Task \n"
L"  -p LPM   Local Port Monitor \n"
L"  mode extensions: \n"
L"  -p                         Same as -pR \n"
L"  -pR <seed>      Seed the pseudo-random key, task, or service name (use for reproducibility) \n"
L"  -pR TIME        Seed with tickcount (DEFAULT) \n"
L"  -pN <name>          Manually specify the name of the key, task, or service (by context) \n"
```

Where to drop an EXE

Where in the registry a key will be made to point to the file*

*the task works a little differently, but you get the idea

Typical Artifacts by Attack Stage

Software Exploitation

- Email, browser, and social media logs
- Downloaded files
- Crashdump files
- Crash events

Malware Deployment

- Email, browser, and social media logs
- Downloaded files
- Secondary files, dropped files
- Persistence
- Injected DLL's or memory
- Non-standard Parent Process
- Non-standard Paths
- Hide in plain sight

C2

- Open connections
- Inet Cookies
- Timer and encryption loops
- User-agent strings

Host Interaction

- Prefetch,Superfetch
- LNK, ShellBags, MRU, UserAssist
- ShimCache
- Index.dat (smb mounts, file share access)
- Ntuser.dat (smb mounts, file share access)
- Event Log

Lateral Access

- Logon Events
- BMC files
- User.dat
- SMB/RDP
- Tasks

Think <u>Created by Software Actions</u>

Potential RangeKit Test Scenarios

Potential Features and Attack Patterns for RangeKit

Attack Stage – Initial Access

- Simulate the injection of shellcode or exploitation of permissions
- Drop <u>artifacts</u> indicative of software exploitation or user behavior with remote interaction or interaction with exploitative device/document/USB/etc
- Simulate bad browsing behavior with exploitation event
  - Inject a crash thread
  - Simulate crashdump files with timestamp
- Due to the way most user apps work, actually exploiting <u>USER apps</u> to create said artifacts would not typically destabilize the environment. However this is NOT the case with server software. <u>Server software should not be exploited in a field environment due to potential destabilization.</u>

Symantec.

Simulate Spear Phishing Attachment

- Attachment
  - Direct execution
  - ICON choice (icon cached on filesystem)
  - Exploitation of client through script (more difficult to emulate)
- Clicked a link, then localhost poison system with server information
  - Add DNS cache
  - Create a closing TIME_WAIT socket in netstat
  - Add entry to browser history / cache
- Simulate an exploit
  - Simulating specific CVE's is a time consuming research project and not sure it matters yet… just any exploit event is enough – why worry about specific bugs when all bugs matter?
  - Inject-and-crash thread

Attachment examples

- Drop Microsoft Word/Excel document with malicious scripts
- Link Fake login portals
- Attach directly a malware (decoy) EXE
- Simulate a user un-RAR / un-ZIP
  - RAR leaves specific artifacts on AppData
- Attach EXE but with tricks to hide extension
  - Use Unicode trick (filename is backwards)

CloudPhishing & Shadow IT

- Same thing, just not email server
- Simulate URL for
  - Google
  - Yahoo
  - Amazon
  - YouTube
  - Skype
  - …….
- Simulate cloud provider services
  - Example, the scenario where you have insider threat moving files via DropBox

I think this is a fruitful area to simulate on an Enterprise-by-Enterprise basis. Each will have specific cloud apps they are worried about / Shadow-IT

Symantec.

Example: Step by Step Simulated Drive By

- Create or use a known bad DNS
- Poison the local DNS cache with this address to point to localhost
- Open port to receive request, simulate interaction
- Inject and crash the browser
- Or, inject and create suspicious dropper behavior from browser
- Clean the DNS cache
- Kill the infected browser
- Cleanup the artifacts after test

Dropped shell, possible behavior…

- Certutil.exe –f
- Cmstp.exe
- Bitsadmin /addfile
- Hh.exe
- Etc… long list… (see LOLBins for extensive list)

Patch Chain/Supply Chain

- Simulate Backdoored Updater
  - Bad/Fake GoogleUpdate.exe
- Non-signed update
- Updater.exe performs anomalous behavior
  - Inject into actual GoogleUpdate.EXE and perform suspicious operations

Removable Media

- Create RegKey for USB device, sync the insert time w/ a file creation time in a common directory (like desktop or LNK)
  - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USBSTOR\
  - \Windows\inf\setupapi.upgrade.log (and similar variants)
    - This will have the date/time of the FIRST time the USB device was inserted (sync point)
- Hijack an existing USB device insertion event
  - Change metadata to simulate malicious activity
- Attempt to add a new log line to this file (not locked on win7)

Symantec™

Attack Stage - Execution

- Software running secondary process, injecting code, running foreign introduced commands or code
- Can execute the secondary behavior through a large number of vectors
- Creating secondary execution does not typically cause destabilization of the environment. However, use of <u>injection is difficult if not impossible to remove after the test</u> – so if you want zero trace testing, you can:
  - Test only injection scenarios on user mode apps that can safely be killed after the test
  - Or, DECOY target programs that look and smell like server software but are actually set up with the sole purpose of being an injection target
  - Injection against actual server processes should not be performed in field unless you are OK leaving the injected payload in place after the test. This can be OK because the injected payload will be INERT, but it can trigger false positives in your security software when scanned.

Symantec.

```
GetModuleFileName(GetModuleHandle(NULL), filename, sizeof(filename));

GetTempPath(MAX_PATH, szTmpPath);
//GetTempFileName(szTmp, L"br", 0, szTmpFile); <-- now using generator
GenerateFileName(key, szTmpFilename);
_snwprintf_s(szTmpFullPath, MAX_PATH - 2, _TRUNCATE, L"%ws\\%ws", szTmpPath, szTmpFilename);

CopyFile(filename, szTmpFullPath, FALSE);
```

Making a copy of oneself to a new, temporary directory.

```
 memset(&process_info, 0, sizeof(process_info));
 memset(&startup_info, 0, sizeof(startup_info));

 WCHAR cmdline[MAX_PATH];
 _snwprintf_s(cmdline, MAX_PATH - 2, _TRUNCATE, L"%ws %d RUN", szTmpFullPath, key);

 CreateProcess(
         szTmpFilename,
         cmdline,
         NULL,
         NULL,
         TRUE,
         DETACHED_PROCESS,
         NULL,
         szTmpPath,
         &startup_info,
         &process_info);

 ExitProcess(EXIT_SUCCESS);
```

Executing the secondary, then exiting the primary.

Simple DECOY signatures

```
// I made this list up - to be fair if you saw these strings in an unknown EXE you would assume it's a RAT -MGH
char gszSuspiciousStrings[] =
" open " \
" download " \
" screenshot " \
" melt " \
" copyfile " \
" upload " \
" delete " \
" keylogger " \
" record " \
" changedns " \
" install " \
" uninstall " \
" stealth " \
" backdoor " \
" rootkit " \
" ";
```

Nothing detects this as malicious.

```
// would trigger fancybear except that this is a PE and that malware uses python
char gszFancyBearLine[] =
"pop.gmail.com\0" \
"jassnovember30@gmail.com\0" \
"30Jass11\0" \
"smtp.gmail.com\0" \
"userdf783@mailtransition.com\0" \
"ginabetz75@gmail.com\0" \
"75Gina75\0" \
"104.152.187.66\0" \
"/updates/\0";

WCHAR gszZeroAccessLine[] =
L"\\Google\\Desktop\\Install\\{%08x-%04x-%04x-%02x%02x-%02x%02x%02x%02x%02x%02x}\\#.\\ \0" \
L"%wS\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\User Shell Folders\0" \
L"Local AppData\0" \
L"\\GLOBAL??\\ \0";
```

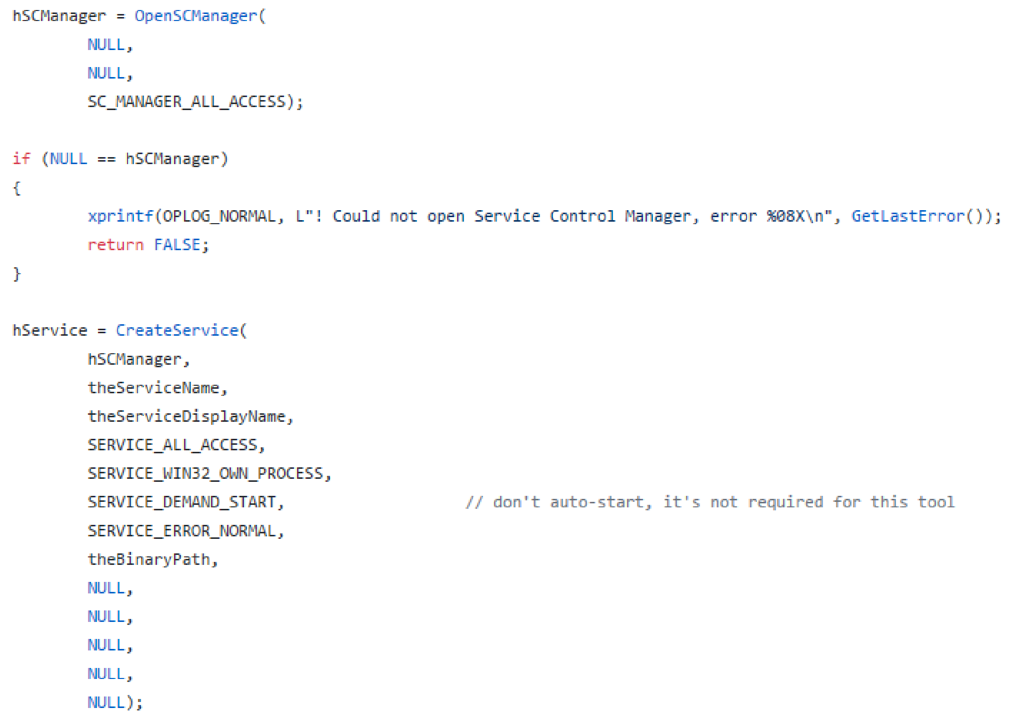
Yet, this is detected as malicious.

Network Coms / C2

- Deliver simulated COMS
- Basic tests
  - Use suspicious DNS or known bad DNS
  - Use suspicious IP or known bad IP
  - Use suspicious protocol/port
- Advanced test
  - Embed COMS within HTTPS
  - Embed COMS within ‘trusted’ cloud traffic (aka chat, cloud storage, etc)
  - Produce poorly encrypted vs strongly encrypted traffic

Attack Stage - Persistence

- Persistence
  - There are so many ways to persist on a Windows box. These are easy to create and then remove later for no-trace testing



```
hSCManager = OpenSCManager(
        NULL,
        NULL,
        SC_MANAGER_ALL_ACCESS);

if (NULL == hSCManager)
{
        xprintf(OPLOG_NORMAL, L"! Could not open Service Control Manager, error %08X\n", GetLastError());
        return FALSE;
}

hService = CreateService(
        hSCManager,
        theServiceName,
        theServiceDisplayName,
        SERVICE_ALL_ACCESS,
        SERVICE_WIN32_OWN_PROCESS,
        SERVICE_DEMAND_START,                   // don't auto-start, it's not required for this tool
        SERVICE_ERROR_NORMAL,
        theBinaryPath,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL);
```

General areas to consider persistence

- Registry keys, tasks, services
- Alternate data streams
- Startup locations, files
- DLL search order
- Javascript, vbscript

Attack Stage – Privilege Escalation

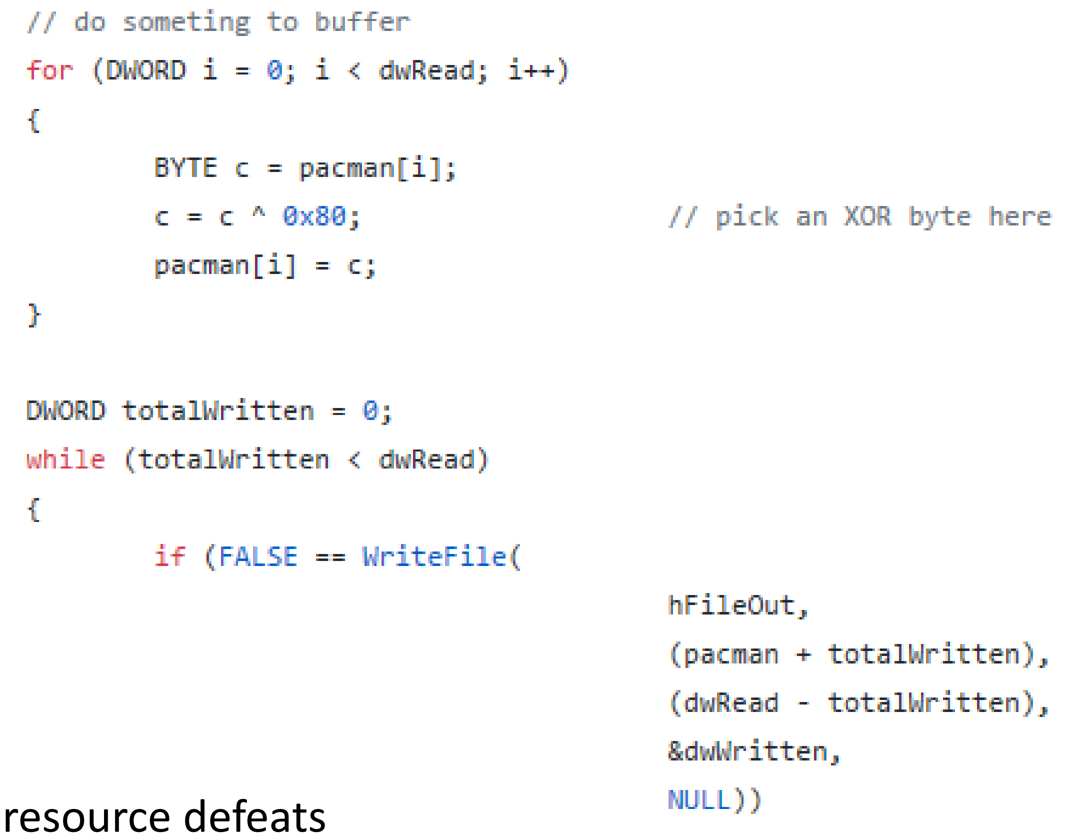
- Privilege Elevation
  - Drop forensic evidence and events that would correspond
  - Or, simply perform the privilege elevation procedure – this would not typically introduce any threat or instability to the environment (may want user to supply accounts to be used in this test)

```
HANDLE hUserToken;
if (FALSE == LogonUser(L"NETWORK SERVICE", L"NT_AUTHORITY", NULL, LOGON32_LOGON_NEW_CREDENTIALS, LOGON32_PROVIDER_WINNT50, &hUserTo
{
        xprintf(OPLOG_NORMAL, L"! Could not elevate privilege, error %08X\n", GetLastError());
        return FALSE;
}

if (FALSE == ImpersonateLoggedOnUser(hUserToken))
{
        xprintf(OPLOG_NORMAL, L"! Could not impersonate user, error %08X\n", GetLastError());
        return FALSE;
```

Attack Stage – Defense Evasion

- Fake code signing
- Binary polymorphism / padding
- Kill services



```
// do someting to buffer
for (DWORD i = 0; i < dwRead; i++)
{
        BYTE c = pacman[i];
        c = c ^ 0x80;                   // pick an XOR byte here
        pacman[i] = c;
}

DWORD totalWritten = 0;
while (totalWritten < dwRead)
{
        if (FALSE == WriteFile(
                                        hFileOut,
                                        (pacman + totalWritten),
                                        (dwRead - totalWritten),
                                        &dwWritten,
                                        NULL))
```

XOR encoding embedded PE resource defeats some scanners.

Attack Stage – Credential Access

- Hash dumping utilities
  - In this case, inject known hashdump utility strings in likely targets like LSASS
  - Or, inject and load lsasrv.dll into a process that normally wouldn’t have it
    - This behavior pattern is indicative of mimikatz
    - Loading the DLL doesn’t cause any problems – it sits dormant
    - Maybe cannot be unloaded (FreeLibraryAndExitThread?)

Attack Stage – Network Discovery / Scanning

- Scanning
  - Just create many TIME_WAIT sockets in the netstat
  - Large sequential ranges

Attack Stage – Lateral Movement

- This one is a little more complicated because you need to simulate artifacts on multiple neighboring computers
- The easiest way to do this is <u>actually perform</u> lateral movement
  - Event log entries are created in this case, which are critical artifacts in this pattern.
  - Performing lateral movement doesn't mean you did anything malicious – so by itself this is fairly inert
  - Event log entries are not simple to remove (no-trace not possible *yet…*)

Advanced Talking Points

Polymorphism / Combinatorics

- Test beyond shallow smoke tests, go for real code coverage
- Polymorphic variations/combinatoric methods of attack patterns
- Change / rotate / mutate most common signature pivots
  - Hash of file(s) (aka file blacklisting)
  - Substring or byte-string matching in-file or in-memory (aka virus signature)

The Good and Bad of Byte or String Patterns

- The most common form of signature
- Not ‘bad’ – it works when you know what you are looking for
  - Used for Known toolkits, Known viruses, and detecting current and present outbreaks of a Known malware (like ‘WannaCry’ for example)
  - People love patterns – as evidenced by customer desire for Yara support in EPP/EDR products
- The problem is that it can be ‘mistakenly’ relied upon for detection new and unknown threats
  - Has near zero value against APT (but to be fair, people know this)
  - The problem starts when it’s the ONLY solution in play

Polymorphisms to Byte or String Pattern

- Byte or string matching, simple modifications
  - Change case of known string signature
    - “Metasploit” becomes “METASPLOIT” or “MeTaSpLoiT”
    - Use of strict byte-pattern match, or uncover overly simplistic string comparisons
  - Move the string or byte pattern
    - 00 00 00 ‘M E T A S P L O I T’ 00 00 00 00 becomes
    - 00 00 00 00 00 00 00 00 00 00 ‘M E T A S P L O I T’ 00 00 00 00 00
    - If pattern is based on seek to specific offset

Mutating File Hashes

- Do so without affecting software / binary functionality
  - Can add bytes to end of file
- To maintain file size
  - Can simply tweak bytes in non-code sections
- Use of decoys
  - Dropped file has known static strings that can be searched out and mutated
  - Does not affect code / binary functionality or file size

Injection Testing

- Injected payloads may be difficult to remove after test
  - You can try de-allocation of the injected memory pages
  - If execution within the page has taken place, there can be no existing references in the host software that points the data or functions in the injected page – this is the core problem that must be avoided if you intend to de-allocate or zero-out the injected page
- OR, use a program that can safely be killed
  - Killing a process cleans up all the memory intrinsically
  - Create the decoy process that will be the target of the injection
  - Or, target a USER process (like the browser or office app)
  - Don’t target actual server processes, use decoys if you need to simulate this

Just a note on Timestamp Simulation

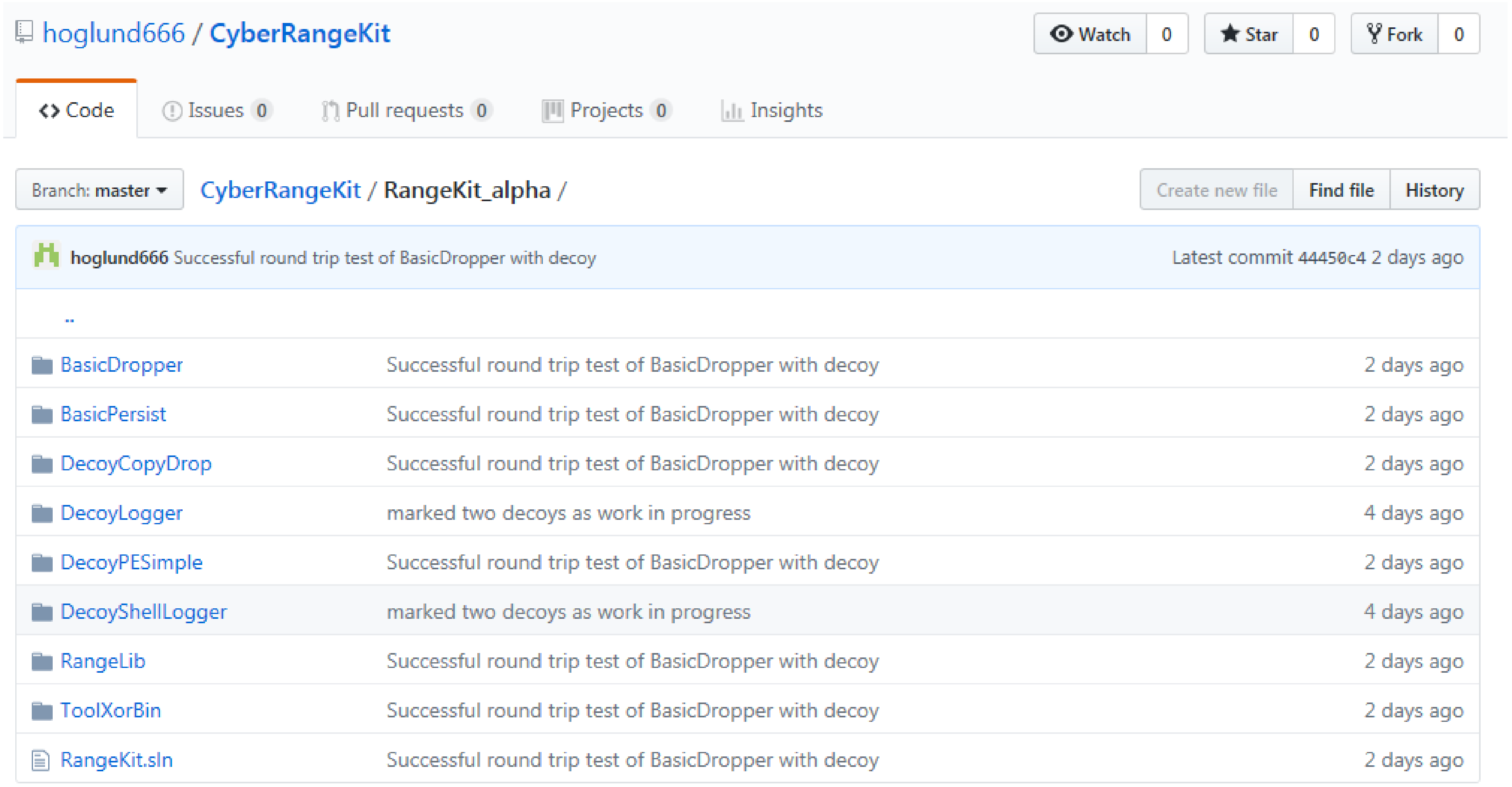
- When simulating timestamps you must pay special attention to whether the artifact is meant to be stored in local system time or UTC. You need to convert all relevant timestamps accordingly.

Lastly…

Symantec™

Where to get it!!

- https://github.com/hoglund666/CyberRangeKit/



hoglund666 / CyberRangeKit

Watch 0 Star 0 Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Insights

Branch: master CyberRangeKit / RangeKit_alpha /

Create new file | Find file | History

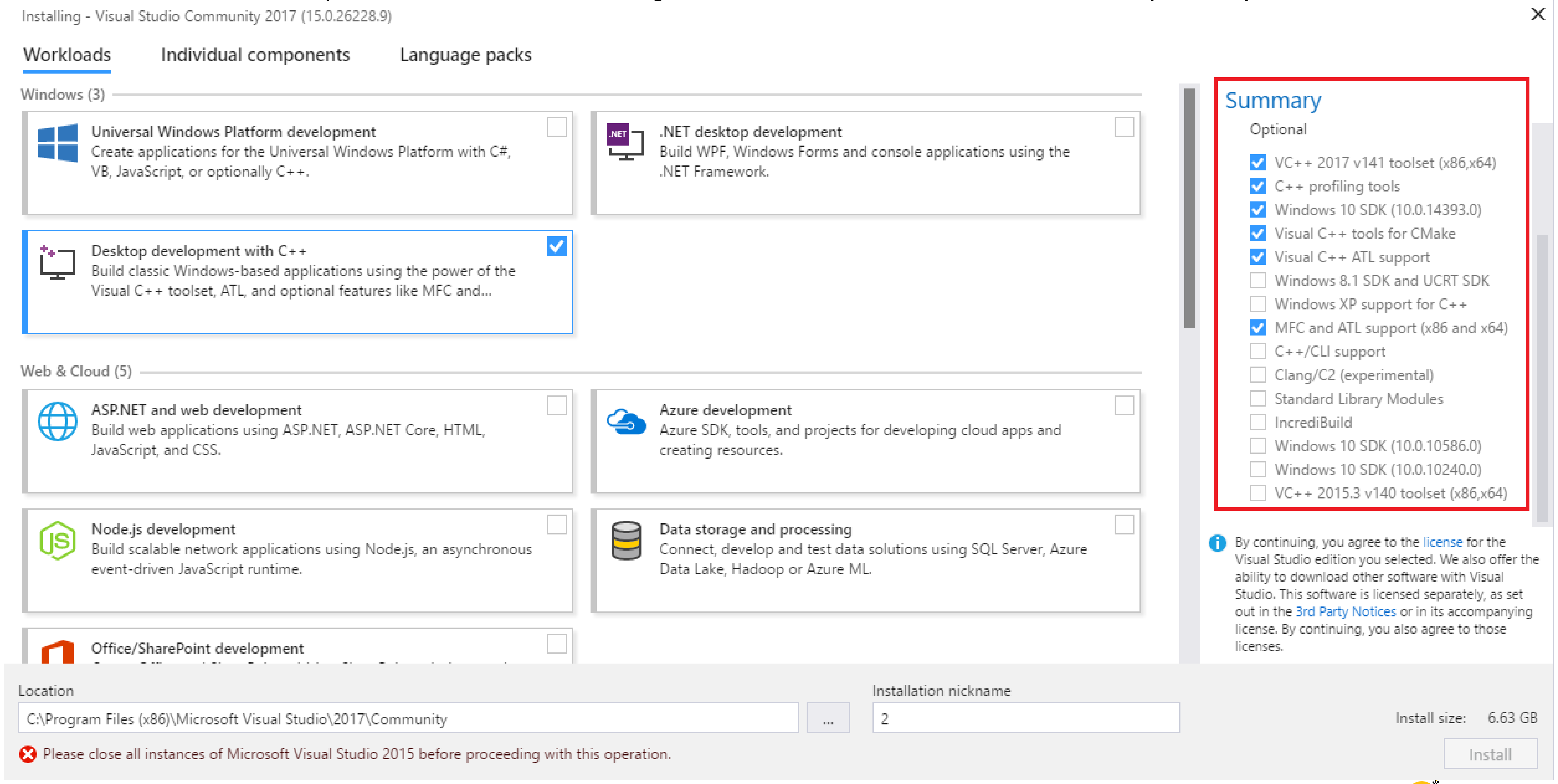
hoglund666 Successful round trip test of BasicDropper with decoy Latest commit 44450c4 2 days ago

..

BasicDropper	Successful round trip test of BasicDropper with decoy	2 days ago
BasicPersist	Successful round trip test of BasicDropper with decoy	2 days ago
DecoyCopyDrop	Successful round trip test of BasicDropper with decoy	2 days ago
DecoyLogger	marked two decoys as work in progress	4 days ago
DecoyPESimple	Successful round trip test of BasicDropper with decoy	2 days ago
DecoyShellLogger	marked two decoys as work in progress	4 days ago
RangeLib	Successful round trip test of BasicDropper with decoy	2 days ago
ToolXorBin	Successful round trip test of BasicDropper with decoy	2 days ago
RangeKit.sln	Successful round trip test of BasicDropper with decoy	2 days ago

Symantec.

I have compiled the various EXE's using SDK 7 with windows XP backwards compatibility… so….


Installing - Visual Studio Community 2017 (15.0.26228.9)

Workloads | Individual components | Language packs

Windows (3)

- [ ] **Universal Windows Platform development** Create applications for the Universal Windows Platform with C#, VB, JavaScript, or optionally C++.
- [ ] **.NET desktop development** Build WPF, Windows Forms and console applications using the .NET Framework.
- [✓] **Desktop development with C++** Build classic Windows-based applications using the power of the Visual C++ toolset, ATL, and optional features like MFC and...

Web & Cloud (5)

- [ ] **ASP.NET and web development** Build web applications using ASP.NET, ASP.NET Core, HTML, JavaScript, and CSS.
- [ ] **Azure development** Azure SDK, tools, and projects for developing cloud apps and creating resources.
- [ ] **Node.js development** Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.
- [ ] **Data storage and processing** Connect, develop and test data solutions using SQL Server, Azure Data Lake, Hadoop or Azure ML.
- [ ] **Office/SharePoint development**

Summary

Optional

- [✓] VC++ 2017 v141 toolset (x86,x64)
- [✓] C++ profiling tools
- [✓] Windows 10 SDK (10.0.14393.0)
- [✓] Visual C++ tools for CMake
- [✓] Visual C++ ATL support
- [ ] Windows 8.1 SDK and UCRT SDK
- [ ] Windows XP support for C++
- [✓] MFC and ATL support (x86 and x64)
- [ ] C++/CLI support
- [ ] Clang/C2 (experimental)
- [ ] Standard Library Modules
- [ ] IncrediBuild
- [ ] Windows 10 SDK (10.0.10586.0)
- [ ] Windows 10 SDK (10.0.10240.0)
- [ ] VC++ 2015.3 v140 toolset (x86,x64)

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Location: C:\Program Files (x86)\Microsoft Visual Studio\2017\Community

Installation nickname: 2

Install size: 6.63 GB

Please close all instances of Microsoft Visual Studio 2015 before proceeding with this operation.

Install

The Future

- Project is in alpha so most of the code will be refactored at some point
- Due to the sheer volume of combinatorics/options it __may__ turn into a single tool that takes a simple scripting language
- I have already moved much of the code into a static library, but this needs to be upgraded to a c++ class and cleaned up (endless TODO)
- Beta will start once I get enough feedback from potential users

How to contribute!

- Test it!
- Give me suggestions on how you want to use it!
- Test it!
- Write some code for it!
- Test it!
- Bug reports, field reports, feature requests, and blue sky!

Thank You!

Greg Hoglund

Please link to me on LinkedIN regarding this project!

Special Thanks to Symantec Corporation for supporting this project!