



Run-Time Detection of Self-Replication in Binary Malware

Alexander Volynkin
Victor Skormin
Douglas Summerville
James Moronski

Binghamton University

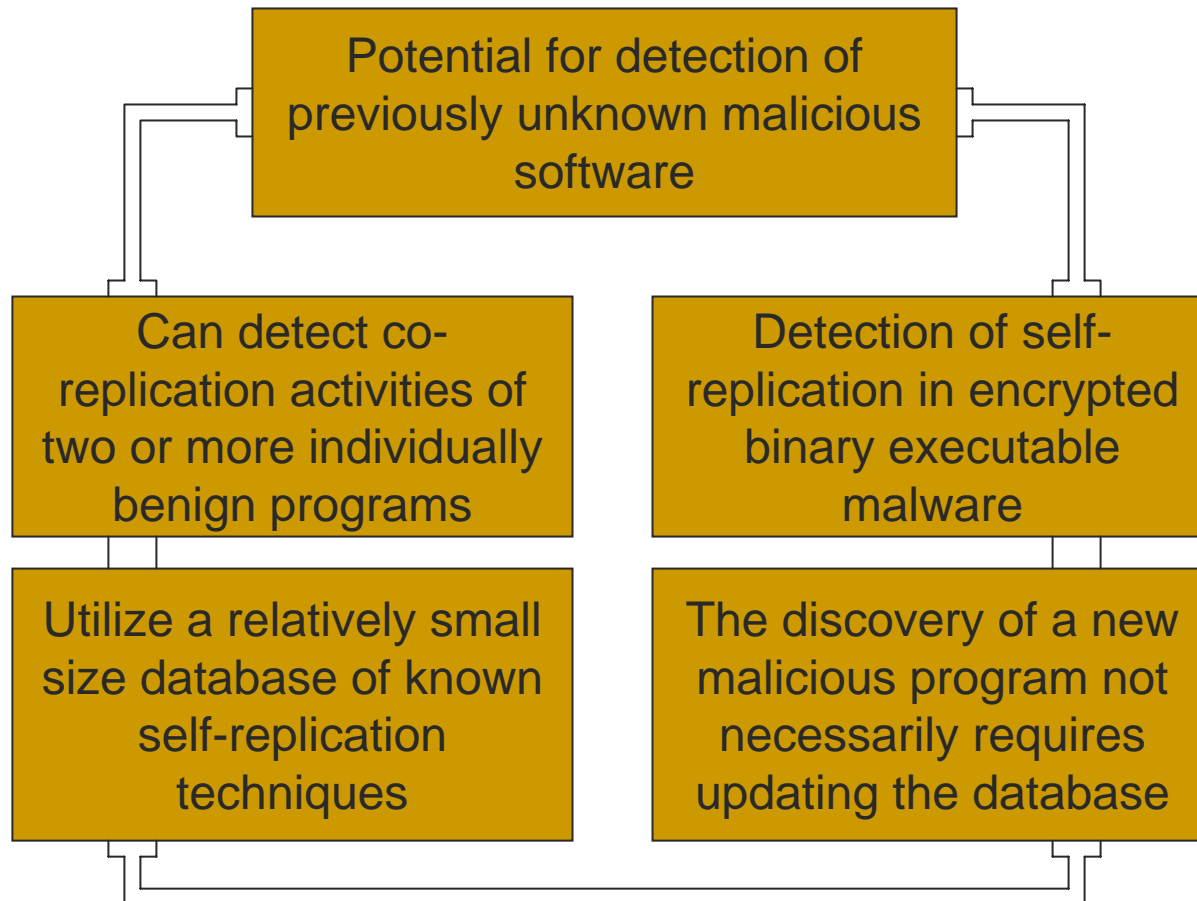
[Gene of Self-Replication]

Published previously

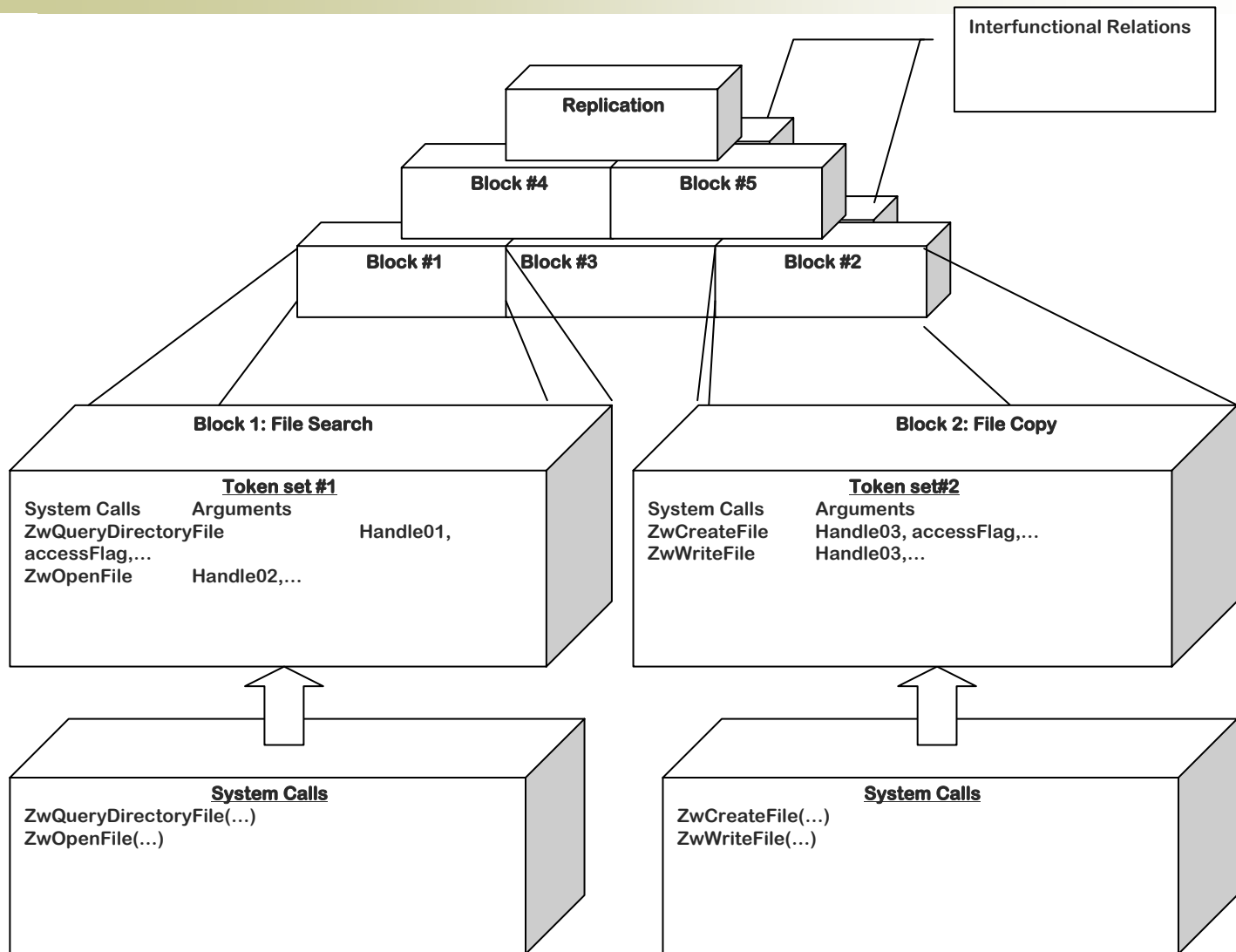
- Malware self-replicates to maximize the impact
- The number of practical techniques to implement self-replication is limited
- Developers of new viruses are destined to rely on a number of existing replication techniques
- Legitimate software seldom self-replicates

Gene of Self-Replication

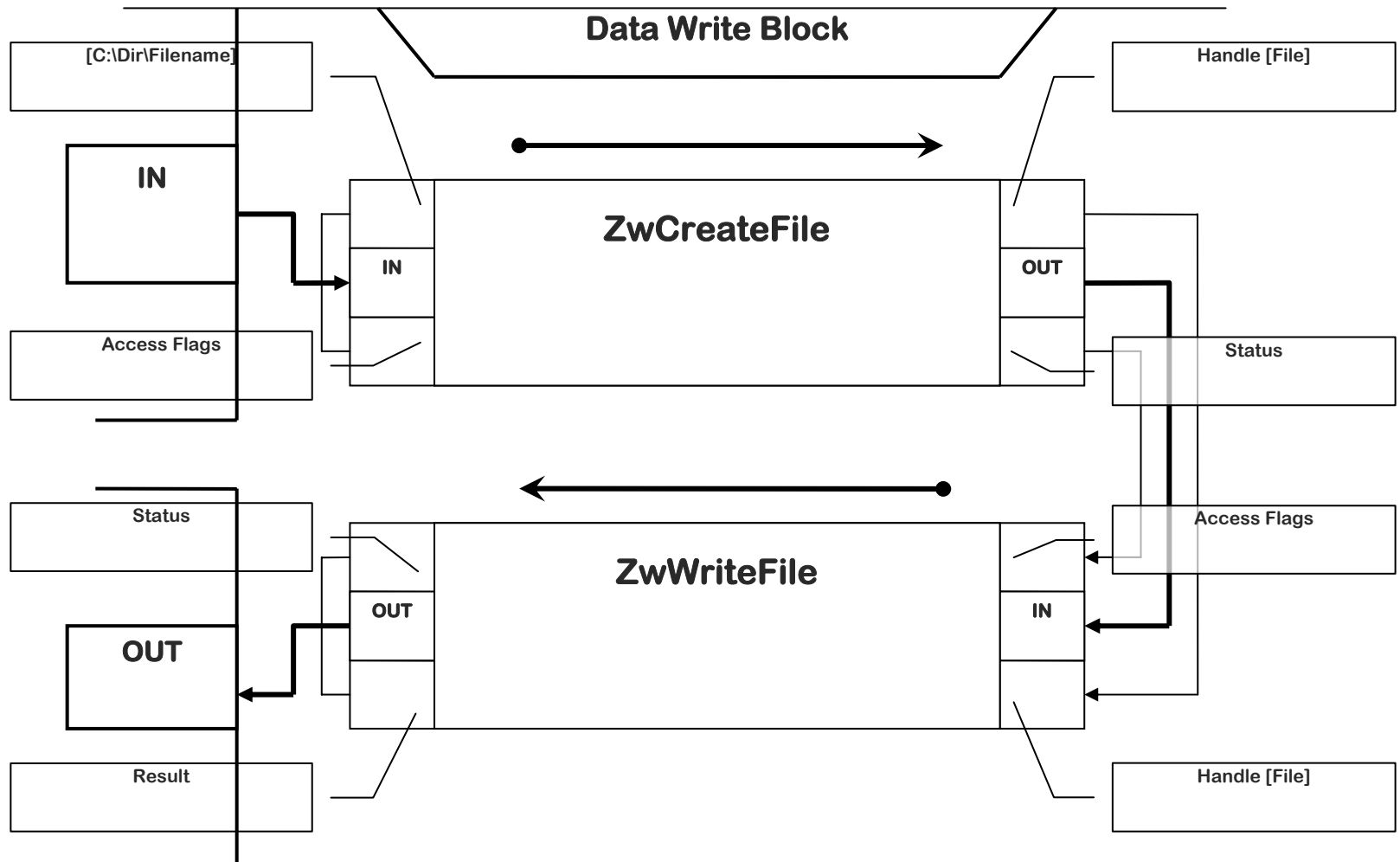
Advantages of our approach:



GSR Structure



GSR Block Structure



Sample Replication Structure

$$G = \{V_N, V_T, P, S\}$$

$$V_N = \left\{ \begin{array}{l} \langle \text{Gene_of_self_replication} \rangle, \langle \text{File_Search_Block} \rangle, \\ \langle \text{File_Copy_Block} \rangle, \langle \text{Directory_System_Call} \rangle, \\ \langle \text{Open_File_System_Call} \rangle, \langle \text{Create_File_System_Call} \rangle, \\ \langle \text{Write_File_System_Call} \rangle \end{array} \right\}$$

$$V_T = \left\{ \begin{array}{l} \text{ZwQueryDirectoryFile(...)}, \text{ZwOpenFile(...)}, \\ \text{ZwCreateFile(...)}, \text{ZwWriteFile(...)} \end{array} \right\}$$

Sample Replication Structure

Gene → *File_Search_Block* · *File_Copy_Block*

File_Search_Block → *Directory_System_Call* · *Open_File_System_Call*

File_Copy_Block → *Create_File_System_Call* · *Write_File_System_Call*

Directory_System_Call → *input₁* · *ZwQueryDirectoryFile* · *output₁*

Open_File_System_Call → *input₂* · *ZwOpenFile* · *output₂*

Create_File_System_Call → *input₃* · *ZwCreateFile* · *output₃*

Write_File_System_Call → *input₄* · *ZwWriteFile* · *output₄*

Sample Replication Structure

$\delta(\text{Gene}, \text{ZwQueryDirectoryFile}) = \{\text{File_Search_Block}\}$

$\delta(\text{Gene}, \text{ZwOpenFile}) = \{\text{File_Search_Block}\}$

$\delta(\text{Gene}, \text{ZwCreateFile}) = \{\text{File_Copy_Block}\}$

$\delta(\text{Gene}, \text{ZwWriteFile}) = \{\text{File_Copy_Block}\}$

$\delta(\text{File_Search_Block}, \text{ZwQueryDirectoryFile}) = \{\text{Directory_System_Call}\}$

$\delta(\text{File_Search_Block}, \text{ZwOpenFile}) = \{\text{Open_File_System_Call}\}$

$\delta(\text{File_Copy_Block}, \text{ZwCreateFile}) = \{\text{Create_File_System_Call}\}$

$\delta(\text{File_Copy_Block}, \text{ZwWriteFile}) = \{\text{Write_File_System_Call}\}$

$\delta(\text{File_Search_Block}, \text{ZwCreateFile}) = \delta(\text{File_Search_Block}, \text{WriteFile}) = 0$

$\delta(\text{File_Copy_Block}, \text{ZwQueryDirectoryFile}) = \delta(\text{File_Copy_Block}, \text{ZwOpenFile}) = 0$

Replication in Malware

Worm Xanax

NtOpenFile 100020h, {24, 0, 42h, 0, 0, "\??\c:\Virlab\"}, 3, 33 ... 12, 0h, 1) result = 0	1
---	---

NtCreateFile 80100080h, {24, 12, 42h, 0, 1243404, "xanax.exe"}, 0h, 128, 3, 1, 96, 0, 0 ... 68, 0h, 1) result = 0	2
---	---

System Call	Input Arguments	Output Args	
NtOpenFile 0x100001	{24, 0, 0x40, 0, 0, "\??\C:\WINDOWS\", 3, 16417	12, {0x0, 1}	3
NtQueryDirectoryFile 12	0, 0, 0, 1243364, 616, 3, 1, "<.exe", 0	{0x0, 11, 0}	4

Replication in Malware

Worm Xanax (cont)

System Call	Input Arguments	Output Args	
NtCreateSection 0xf001f	0h, 0h, 2, 134217728, 68	72, 5	Virus Handle
NtMapViewOfSection	-1, 0h, 0, 0, {0, 0}, 0, 1, 0, 2	0x980, 000, 0, 0, 368, 64	Section Handle
NtCreateFile 0x40110080	{24, 0, 40h, 0, 1242788, "\??\C:\WINDOWS\calc.exe"}, 0h, 32, 0, 5, 100, 0, 0	52, {0h, 3}	Victim File
NtSetInformationFile 52	1241948, 8, 20	{0h, 0}	End Of File
NtWriteFile 52	0, 0, 0, "MZ\220\0\3\0\0\0\4\0\0\0\377\37.....\0\0\0", 33792, 0h, 0	{0h, 33792}	Viral Code Code Size

Virus Replication Data

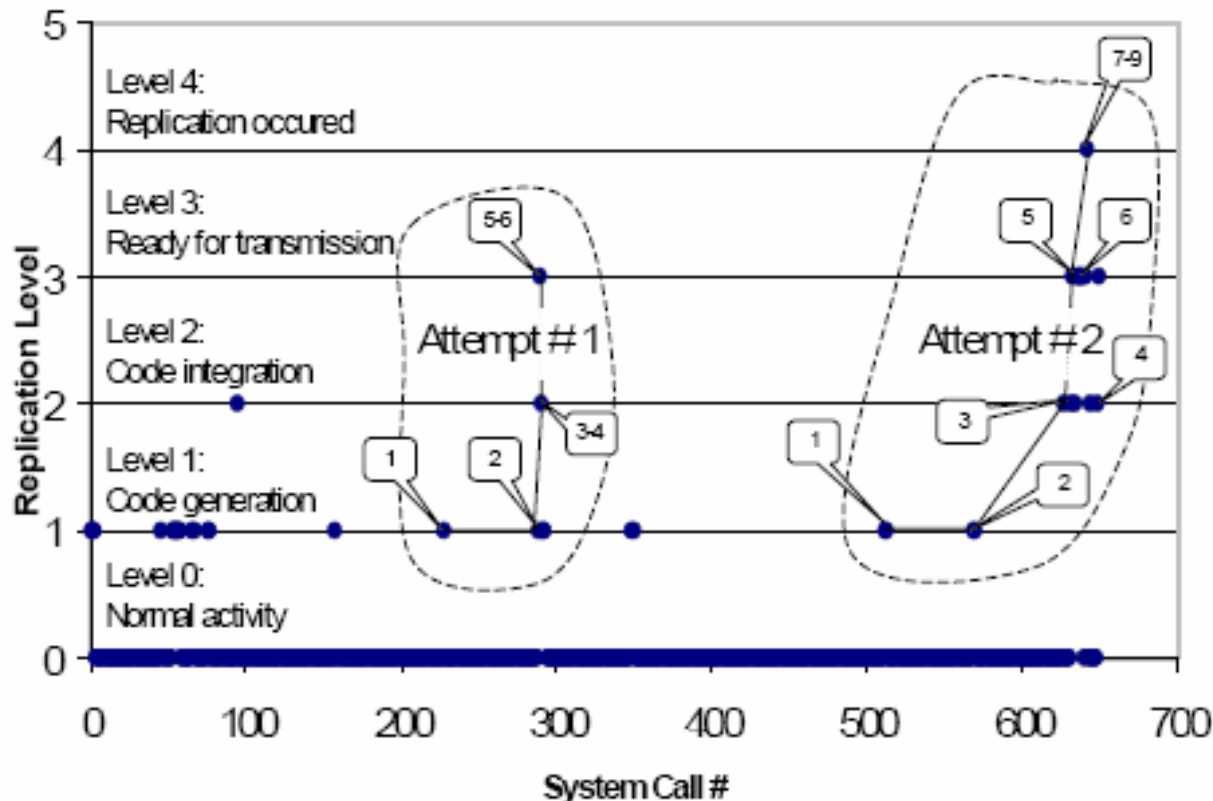


Fig. 4. Sample Virus Replication Data (648 points, 2 attempts).

Extra Features: Weights

Each individual GSR component is assigned with a weight

$$\begin{aligned} W_B = & \sum (W_{B_1}, W_{B_2}, W_{B_3}, \dots, W_{B_n}) + \\ & + \sum (W_{B_{bind(1 \leftrightarrow 2)}}, W_{B_{bind(2 \leftrightarrow 3)}}, W_{B_{bind(3 \leftrightarrow 4)}}, \dots, W_{B_{bind(n-1 \leftrightarrow n)}}) + \\ & + \sum (W_{B_{1in}}, W_{B_{1out}}, W_{B_{2in}}, W_{B_{2out}}, \dots, W_{B_{nin}}, W_{B_{nout}}) + W_{Result} \end{aligned}$$

Normalized Replication Score is computed as follows:

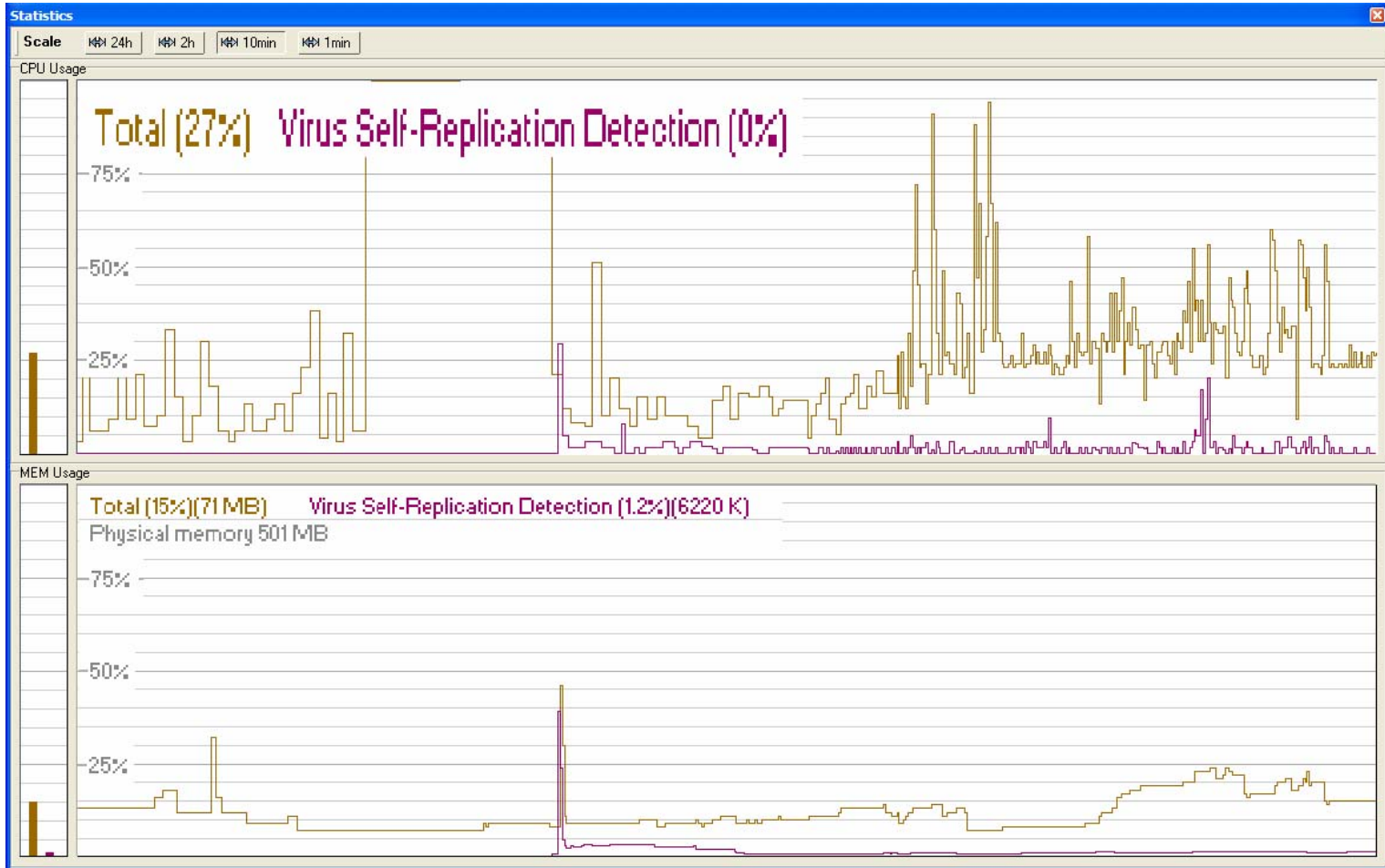
$$R_{norm} = \frac{(R + \sum (W_{rB_1}, W_{rB_2}, \dots, W_{rB_n}))}{N} \cdot 100\%$$

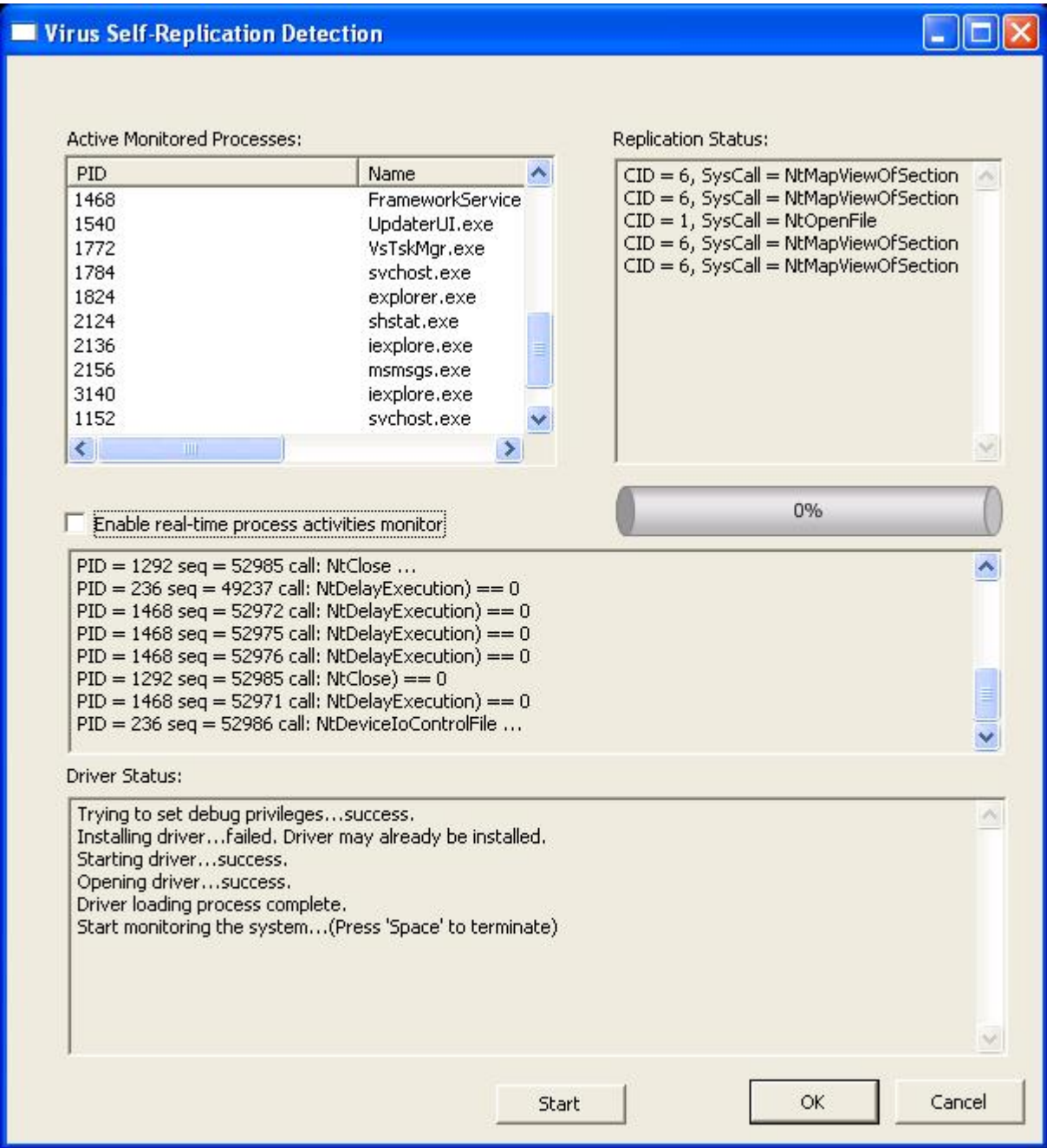
Replication Rates

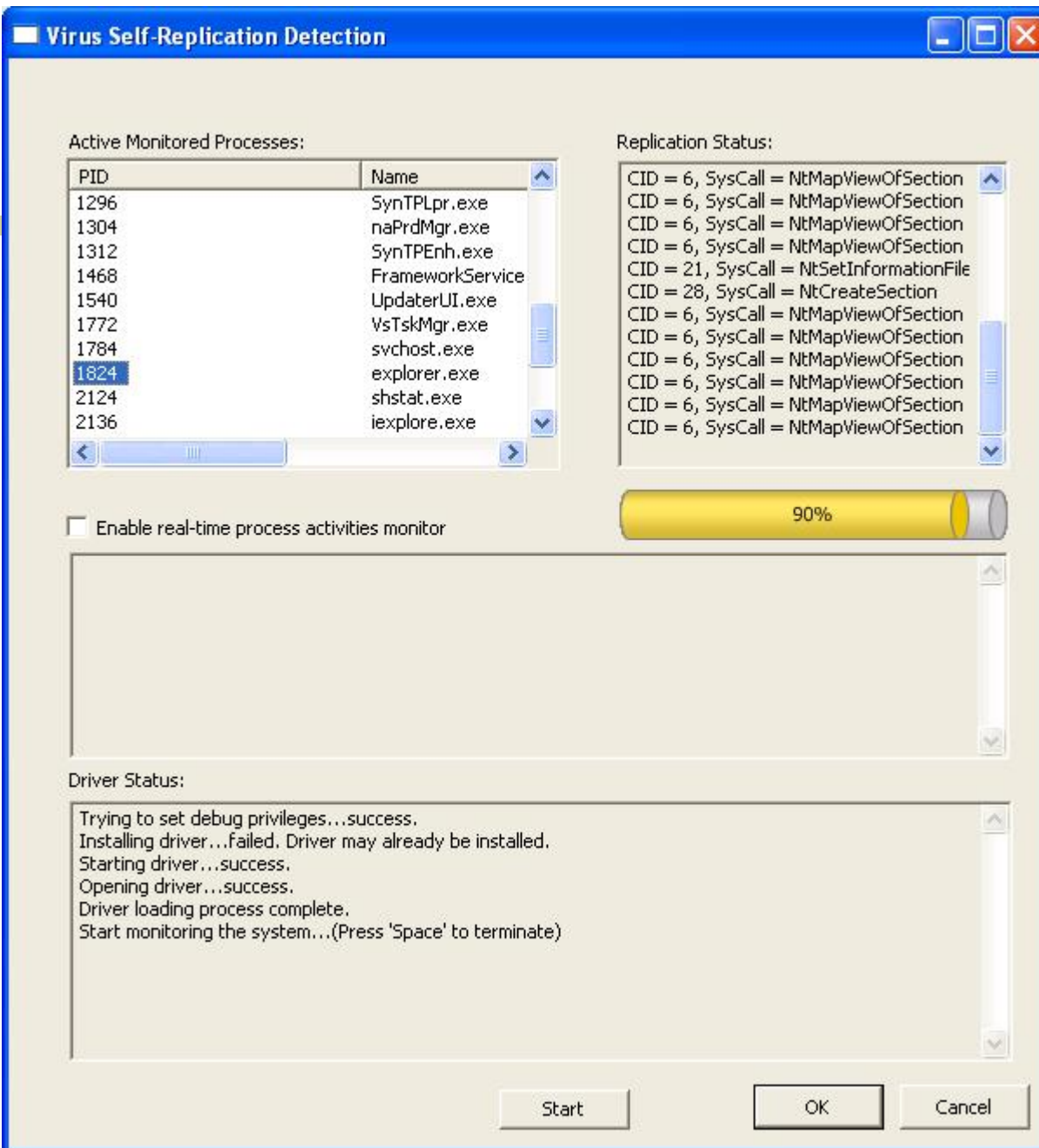


	Host Search	File Access	Networking	Memory	Injection / infection	Normalied Replication (total)
W32.Alicia	100%	100%	100%	32.4%	100%	100%
W32.Bogus	100%	100%	5.3%	3.7%	100%	100%
W32.Crash	100%	100%	0%	100%	100%	100%
W32.Neo	100%	100%	7.0%	100%	100%	100%
W32.Linda	100%	100%	4.3%	100%	100%	100%
W32.Stream	100%	100%	32.5%	100%	100%	100%
Svchost.exe	26.3%	100%	79.4%	100%	36.0%	78.4%
Explorer.exe	14.5%	92.1%	100%	84.5%	47.4%	86.2%

Overheads







Virus Self-Replication Detection

Active Monitored Processes:

PID	Name	Status	Time
1676	explorer.exe		Created: 07
1848	VMwareUser....		Created: 07
1832	VMwareTray....		Created: 07
4	System		Created: 07
700	savedump.exe		Created: 07
708	lsass.exe		Created: 07
432	savior.EXE	Suspended	Created: 07
644	winlogon.exe		Created: 07
876	svchost.exe		Created: 07
1864	msmsgs.exe		Created: 07

Replication Status:

CID = 6, SysCall = NtMapViewOfSection
CID = 6, SysCall = NtMapViewOfSection
CID = 27, SysCall = NtWriteFile
CID = 10, SysCall = NtWriteFile
CID = 27, SysCall = NtWriteFile
CID = 10, SysCall = NtWriteFile
CID = 24, SysCall = NtQueryVolumeInfor
CID = 6, SysCall = NtMapViewOfSection
CID = 6, SysCall = NtMapViewOfSection
CID = 25, SysCall = NtQueryInformation
CID = 25, SysCall = NtQueryInformation
CID = 25, SysCall = NtQueryInformation

Enable real-time process activities monitor



PID = 1676 seq = 52526 call: NtReleaseSemaphore ...
PID = 1676 seq = 52522 call: NtOpenKey) == 0
PID = 1676 seq = 52526 call: NtReleaseSemaphore) == 0
PID = 1676 seq = 51458 call: NtUserWaitMessage) == 1
PID = 1676 seq = 52527 call: NtQueryKey ...
PID = 1676 seq = 52528 call: NtUserPeekMessage ...
PID = 1676 seq = 52529 call: NtWaitForSingleObject ...
PID = 1676 seq = 52528 call: NtUserPeekMessage

Driver Status:

Replication Detected!
Process ID: 432
File Name: not available
Replication ID: 129

Start OK Cancel

[Conclusion]

- Monitoring and analysis of system calls at runtime is an affordable technology providing unambiguous insight into what the software actually does, including the self-replication indicative of malicious behavior.
- System calls analysis must include arguments analysis for correct behavior detection
- More work to be done to protect the detector
- Additional replication schemes may be introduced for new virus concepts
- Correct GSR definition is the key for keeping false positives down

[Questions]

Thank you!