

Computer Virus Advancement

John Lynch

Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

November 12, 2016
Morris, Minnesota

Abstract

Why would we make a virus?

We work at some company that we know isn't the greatest.

Long hours, little pay, and Karen keeps taking your meals out of the fridge.

It's time for payback with corporate sabotage.

Please don't actually do this. This is an example with comedy.

Outline

- 1 Introductions
- 2 Applications
- 3 Security
- 4 Conclusions

Background

- Malware: Software that is created for malicious purposes against computer systems.
- Computer Virus: One form of malware that self-replicates in a system
- Genetic Algorithms: A method for solving optimization by mimicking biological evolution.
- Anti-Malware: Software developed to combat malicious software.

Outline

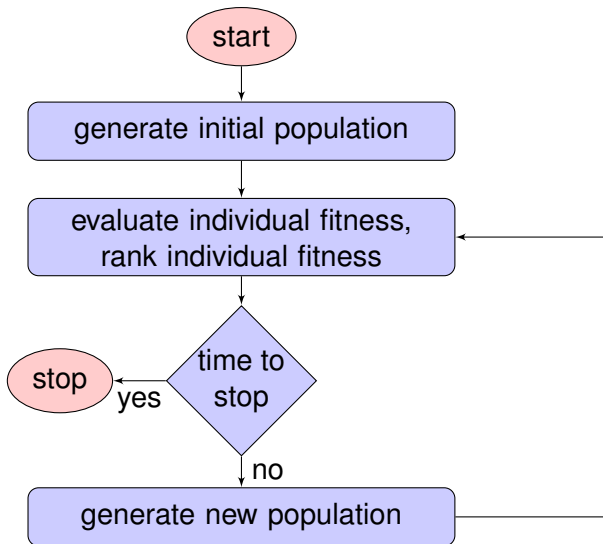
- 1 Introductions
 - Genetic Algorithms
 - Computer viruses

- 2 Applications

- 3 Security

- 4 Conclusions

What it looks like



method of evolution presented by Thomas Back

Basic virus structure

- parts
 - trigger
 - payload
 - infection mechanism
- phases
 - dormant phase
 - propagation phase
 - trigger phase
 - execution phase

Our Virus

- supahVirus.bat
- :swarmMeth
- :mechMeth
- goal

Outline

1 Introductions

2 Applications

- Hiding in plain sight
- Faster from training

3 Security

4 Conclusions

How :swarmMeth works

- This is used during propagation phase.
- Return the copy of the virus.
- In Batch this would be creating a new terminal with the same function.

Simple :swarmMeth

```
swarmMeth() {  
  initialize(thisVirus);  
  while true do {  
    return copy(thisVirus);  
  }  
}
```

Improving :swarmMeth

```
swarmMeth() {  
  initialize(thisVirus);  
  while true do {  
    mutatedVirus = mutate(thisVirus);  
    if mutatedVirus.name != thisVirus.name {  
      initialize(mutatedVirus);  
    }  
  }  
}
```

Names to change

Supahvirus.bat

Superbvirus.bat

Su9@virunala.bat

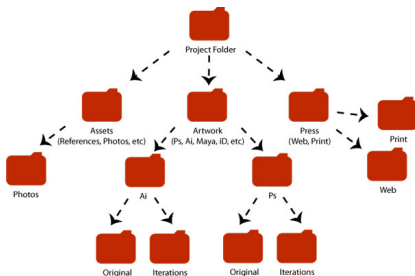
youGetTheIdea.bat

Heuristic search

- Where do we want to search first?
- What folders are more likely to have what we're looking for?
- We need to sort out priorities.

Training our search

- Begin training in offline search environments.
- Randomly create folders that simulate company computer file structures.
- Keep track of folder structures that consistently appear, use those common occurrences.



method of training presented by Sadia Noreen et. al. of next generation intelligent network research

How :mechMeth works

- This is the infection mechanism.
- We move each new copy that we make with :swarmMeth

Simple :mechMeth

```
mechMeth () {
  while true do{
    for each folder f in folders {
      if (folder.name == "secretFolder"){
        copyAndTransfer(f);
      }else{
        into (swarmMeth, f);
      }
    }
  }
}
```

Improving :mechMeth

```
mechMeth () {  
  folders = new priorityMap[heuristic]  
  while true do{  
    for each folder f in folders {  
      if (folder.name == "secretFolder"){  
        copyAndTransfer (f)  
      }else{  
        swarmMeth()  
        into (swarmMeth.mutatedVirus , f)  
        initialize (mutatedVirus)  
      }  
    }  
  }  
}
```

What we can genetically modify

- Names, each character that is put into the name of the virus.
- Payload, different pictures or modifications to files.
- Variations of methods, changing what the methods do and how they do it.

Outline

1 Introductions

2 Applications

3 Security

- Detecting machine generated malware
- Different bases of defense

4 Conclusions

Basic anti-malware

- Does it have a bad name?
- Why do these programs have the same name in process and action?
- Where did the processing power go?
- If you see something, say something.



McGruff the crime dog. all rights reserved.

Anti-malware based on signatures

- What programs running are doing the same thing?
- Are the processes using similar power?
- Are there similar code structures that reappear?
- Where did all these processes come from?
- Different name, but we know it's the same game.



Signature detection based on research by Kandissounon and Chouchane of Columbus University

Defenses

User diligence

- Can the user see when and where there is a malicious process?
- How long until a user can take action?
- Does a user have the power to override ongoing processes?

Anti-malware diligence

- When does the anti-malware notice something is amiss?
- Does the anti-malware throttle processes that seem malicious?
- Can the anti-malware defend against its own destruction?
- Can the anti-malware alert the user?

Defenses presented by Yang Wang and Chenxi Wang, Carnegie Mellon University

Outline

1 Introductions

2 Applications

3 Security

4 **Conclusions**

- Results
- What we've learned

Results

Several studies have been conducted on evolving malware and anti-malware. Both malware and anti-malware are improvable by these processes.

A study introduced evolving malware with basic and signature-based anti-malware. Evolution based programs significantly improved the efficiency of both malware and anti-malware.

See references for more details on exactness of studies. Sadia Noreen et. al.

What we've learned

Evolutionary computation may be the cornerstone of improvement to malware and anti-malware as we make advances in computer science.

Each new generation of viruses and anti-malware will only become stronger in their efforts to accomplish whatever goals they set.

As a virus is constructed to spread they gain strength by more than one utility at their disposal.

Thanks!




Thank you for your time and don't do what I just talked about. I may have made several watch-lists. Special thanks to Elena Machkasova, Kristin Lamberty, Nic Mcphee, and my reviewer for putting up with me.

Contact:



`lynch446@morris.umn.edu`

Questions?

References

-  Sadia Noreen, et. al. Evolvable Malware, GECCO '09 Proceedings of the 11th Annual conference on Genetic and evolutionary computation Pages 1569-1576, Canada
-  Yang Wang, Chenxi Wang, Modeling the Effects of Timing Parameters on Virus Propagation, WORM '03 Proceedings of the 2003 ACM workshop on Rapid malcode Pages 61-66, Washington, DC
-  Andrea Cani, et. al. Towards Automated Malware Creation: Code Generation and Code Integration, SAC '14 Proceedings of the 29th Annual ACM Symposium on Applied Computing Pages 157-160, Torino, Italy

References

-  Kandissounon, Chouchane, A Method for Detecting Machine-generated Malware, ACM-SE '11 Proceedings of the 49th Annual Southeast Regional Conference, Kennesaw, Georgia
-  Thomas Back, Evolution strategies: basic introduction, GECCO '13 Companion Proceedings of the 15th annual conference companion on Genetic and evolutionary computation Pages 265-292, New York, NY

See my senior seminar paper for additional references.