11TH INTERNATIONAL CONFERENCE ON

ELECTRICAL AND COMPUTER ENGINEERING (ICECE)

December 17-19, 2020 BUET, Dhaka, Bangladesh icece.buet.ac.bd Conference Record Number: #51571

PAPER ID 828

Static Detection of Malicious Code in Programs Using Semantic Techniques

Syed Zami-Ul-Haque Navid, Protik Dey, Shamiul Hasan, and
Muhammad Masroor Ali
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka, Bangladesh

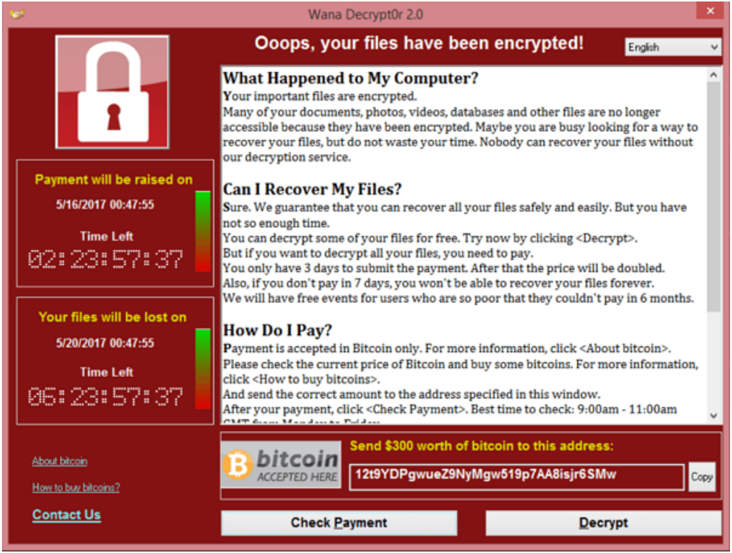
Outline

1. Introduction
2. Objectives
3. Current Trends
4. Methodology
5. Experiments and Results
6. Conclusion and Fututre Works

Outline

1. Introduction
2. Objectives
3. Current Trends
4. Methodology
5. Experiments and Results
6. Conclusion and Fututre Works

Uh oh...



Ooops, your files have been encrypted!

Payment will be raised on

5/16/2017 00:47:55

Time Left

02:23:57:37

Your files will be lost on

5/20/2017 00:47:55

Time Left

06:23:57:37

What Happened to My Computer?

Your important files are encrypted.

Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service.

Can I Recover My Files?

Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time.

You can decrypt some of your files for free. Try now by clicking <Decrypt>.

But if you want to decrypt all your files, you need to pay.

You only have 3 days to submit the payment. After that the price will be doubled.

Also, if you don't pay in 7 days, you won't be able to recover your files forever.

We will have free events for users who are so poor that they couldn't pay in 6 months.

How Do I Pay?

Payment is accepted in Bitcoin only. For more information, click <About bitcoin>.

Please check the current price of Bitcoin and buy some bitcoins. For more information, click <How to buy bitcoins>.

And send the correct amount to the address specified in this window.

After your payment, click <Check Payment>. Best time to check: 9:00am - 11:00am GMT from Monday to Friday.

Send $300 worth of bitcoin to this address:

12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw

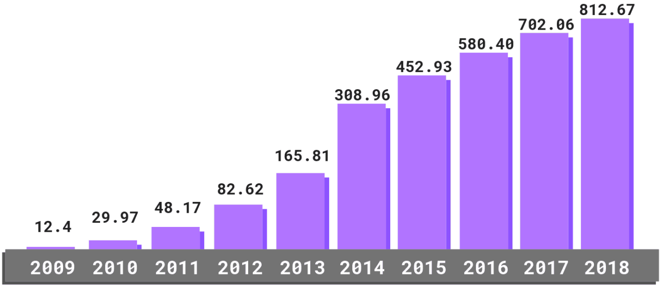
Tell Us Something

What is Malicious Code?

Malicious Code

- Code in any part of a software system or script that is intended to cause,
    - undesired effects,
    - security breaches, or,
    - damage to a system.

Statistics



Total Malware Infection Growth Rate (In Millions)

Octopus Scanner

Outline

1 Introduction

2 Objectives

3 Current Trends

4 Methodology

5 Experiments and Results

6 Conclusion and Fututre Works

Our Goal

- Identifying malware embedded in source code without having to execute the code.

Outline

1 Introduction

2 Objectives

3 Current Trends

4 Methodology

5 Experiments and Results

6 Conclusion and Fututre Works

Current Trends

- There are about two ways of detecting malicious code:

Current Trends

- There are about two ways of detecting malicious code:
    - Dynamic Detection

Current Trends

- There are about two ways of detecting malicious code:
  - Dynamic Detection
  - Static Detection

Dynamic Detection

How To

- The suspected malware is executed in a closely monitored **sandboxed** environment.

Dynamic Detection

How To

- The suspected malware is executed in a closely monitored **sandboxed** environment.

Pitfall

- Despite the sandboxed environment, one still runs the risk of infecting one's system with the malware.

Static Detection

How To

- The most commonly employed process leverages information such as control-API graph and crosschecks against a predefined security policy to give a verdict.

Static Detection

How To

- The most commonly employed process leverages information such as control-API graph and crosschecks against a predefined security policy to give a verdict.

Pitfall

- The security policies themselves can be compromised.

Machine Learning Approach

How to

- The modern machine learning and deep learning approaches make use of neural networks such as CNN, GCN, RNN etc.

Machine Learning Approach

How to

- The modern machine learning and deep learning approaches make use of neural networks such as CNN, GCN, RNN etc.

Pitfall

- These models require huge datasets and demanding processing power which leads to substantial preprocessing and computing time.

Outline

1. Introduction
2. Objectives
3. Current Trends
4. Methodology
5. Experiments and Results
6. Conclusion and Fututre Works

Novelty of Our Work

- As we can see, the aforementioned methods of detecting malicious code require,
  - the inspection of executables, or
  - a predefined security policy, or
  - huge datasets and computation time.

Novelty of Our Work

- As we can see, the aforementioned methods of detecting malicious code require,
  - the inspection of executables, or
  - a predefined security policy, or
  - huge datasets and computation time.
- We eliminate these requirements by introducing **ontology** in this domain.

Novelty of Our Work

- As we can see, the aforementioned methods of detecting malicious code require,
  - the inspection of executables, or
  - a predefined security policy, or
  - huge datasets and computation time.
- We eliminate these requirements by introducing **ontology** in this domain.
- We probe the source code and perform semantic identification of malicious code.

Ontology

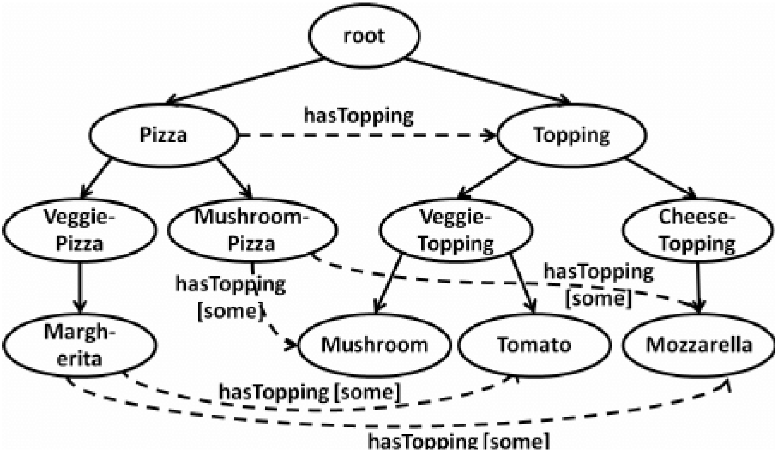
- In computer science and information science, an ontology encompasses,
    - a representation of the categories, properties,
    - relations between the concepts, data and entities.

Components of Ontology

Components

- There are four components in an ontology,
    - Class
    - Object Property
    - Data Property
    - Individuals

Example Ontology



Resources Used

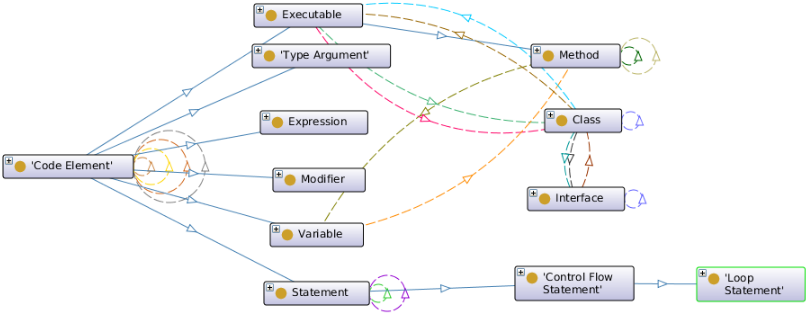
Protege: An open source ontology editor and knowledge management system.

Resources Used

Protege: An open source ontology editor and knowledge management system.

Java Code Ontology: An ontology illustrating the relationships amongst the building blocks of Java programming language.

Code Ontology



Methodology

Steps

Methodology

Steps

- Identifying signatures by studying the source code.

Methodology

Steps

- Identifying signatures by studying the source code.
- Incorporating ontology classes corresponding to signatures.

Methodology

Steps

- Identifying signatures by studying the source code.
- Incorporating ontology classes corresponding to signatures.
- Establishing relationships among the signatures.

Methodology

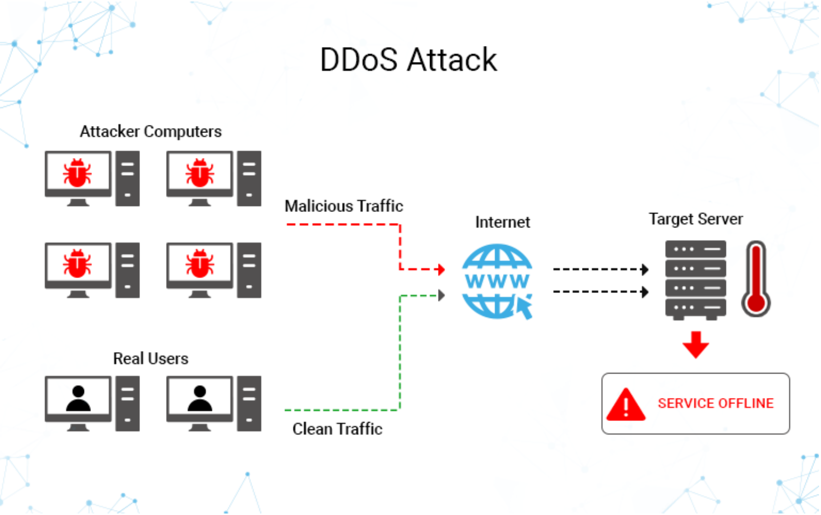
Steps

- Identifying signatures by studying the source code.
- Incorporating ontology classes corresponding to signatures.
- Establishing relationships among the signatures.
- Relating the signature classes to themselves to counteract code obfuscation.

Identifying Signatures

- As an example attempt, we have applied our methodology on the source code of DDoS attack.

DDoS Attack


DDoS Attack
Attacker Computers
Malicious Traffic
Internet
www
Target Server
Real Users
Clean Traffic
SERVICE OFFLINE

Identifying Signatures

Signature Type 1

- Thread Class
- openConnection Method
- setRequestMethod Method

Identifying Signatures

Signature Type 1

- Thread Class
- openConnection Method
- setRequestMethod Method

Signature Type 2

- Thread Class
- Socket Class
- DataOutputStream Method

Creating Corresponding Ontology Classes

Imports

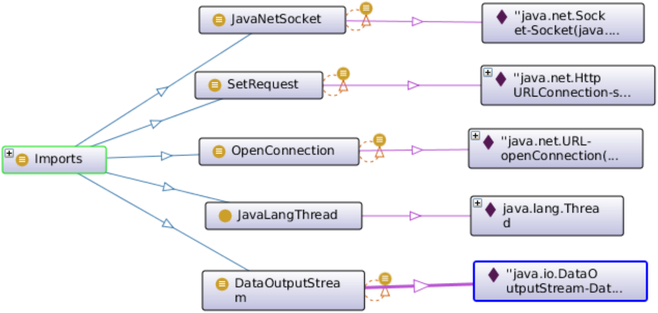
- We create an ontology class named **Imports**.
- As its subclasses we create the following:

Creating Corresponding Ontology Classes

Imports

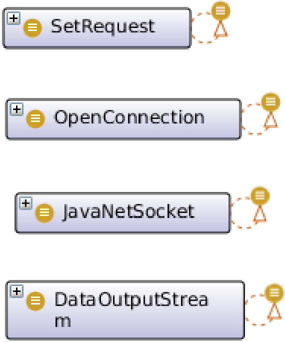
- We create an ontology class named **Imports**.
- As its subclasses we create the following:
  1. JavaLangThread
  2. OpenConnection
  3. SetRequest
  4. JavaNetSocket
  5. DataOutputStream
- These classes represent library methods shipped with Java API.

Imports


Imports
JavaNetSocket
SetRequest
OpenConnection
JavaLangThread
DataOutputStream
"java.net.Socket-Socket(java....
"java.net.HttpURLConnection-s...
"java.net.URL-openConnection(...
java.lang.Thread
"java.io.DataOutputStream-Dat...

Imports

- We assert that, any method that contains calls to a signature method will also be an individual of the said signature method.
- This relationship is expressed through an object property called *references*.

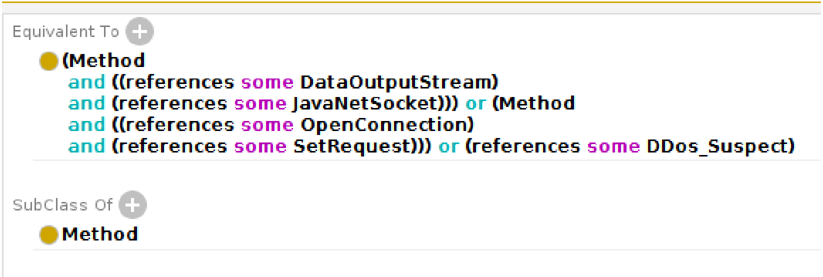


Creating Corresponding Ontology Classes

DDoS_Suspect

- It is a subclass of the ontology class *Method*.
- This *references* either of the signature types.
- DDoS_Suspect also references itself.

DDoS_Suspect



Equivalent To

```
● (Method
   and ((references some DataOutputStream)
   and (references some JavaNetSocket))) or (Method
   and ((references some OpenConnection)
   and (references some SetRequest))) or (references some DDos_Suspect)
```

SubClass Of

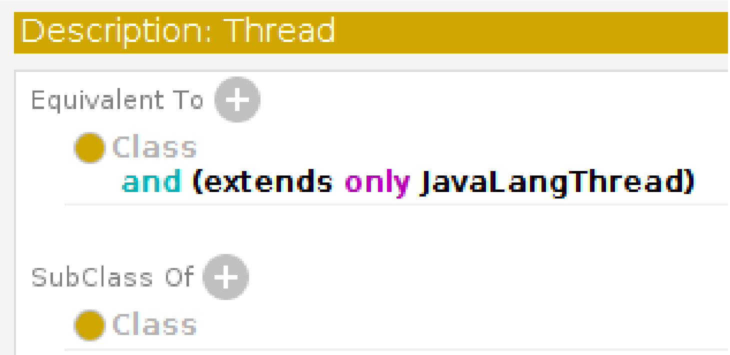
● Method

Creating Corresponding Ontology Classes

Thread

- It is a subclass of the ontology class *Class*.
- It *extends JavaLangThread*.

Thread



Description: Thread

Equivalent To ⊕

● Class
and (extends only JavaLangThread)

SubClass Of ⊕

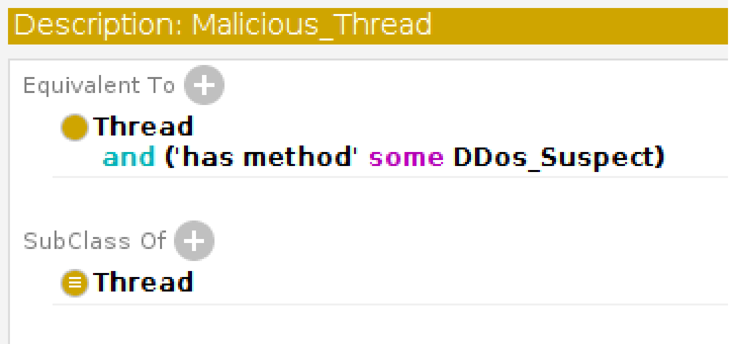
● Class

Creating Corresponding Ontology Classes

Malicious_Thread

- It is a subclass of the ontology class *Thread*.
- It has an instance of *DDoS_Suspect* as one of its methods.

Malicious_Thread



Description: Malicious_Thread

Equivalent To +

Thread
and ('has method' some DDos_Suspect)

SubClass Of +

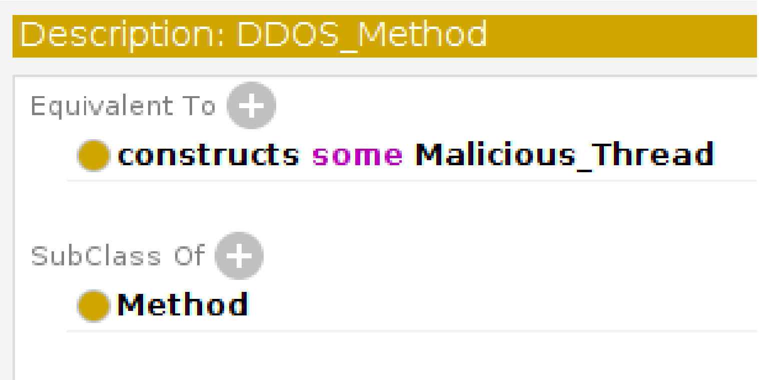
Thread

Creating Corresponding Ontology Classes

DDoS_Method

- It is a subclass of the ontology class *Method*.
- It *constructs Malicious_Thread*.

DDoS_Method



Description: DDOS_Method

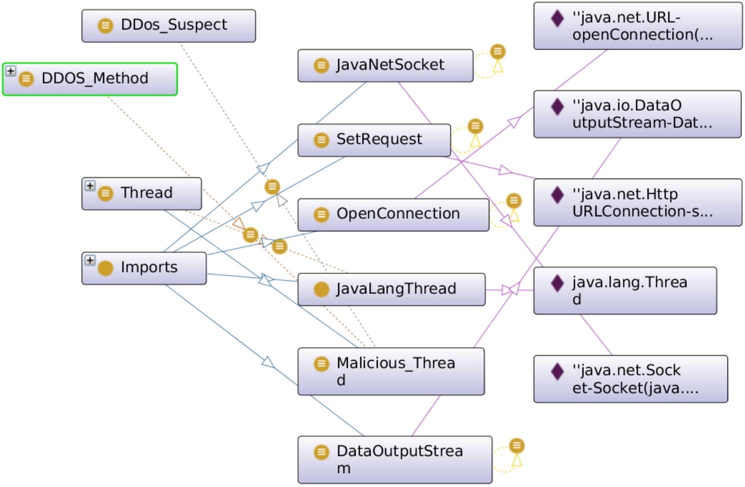
Equivalent To

● constructs some Malicious_Thread

SubClass Of

● Method

Creating Corresponding Ontology Classes



Outline

1 Introduction

2 Objectives

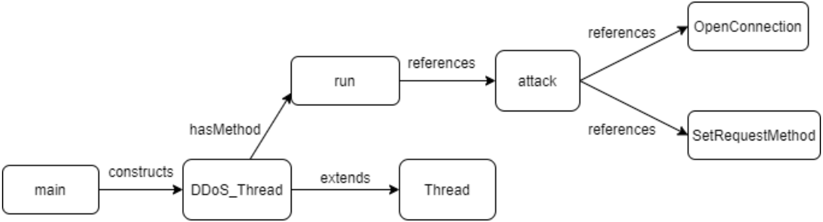
3 Current Trends

4 Methodology

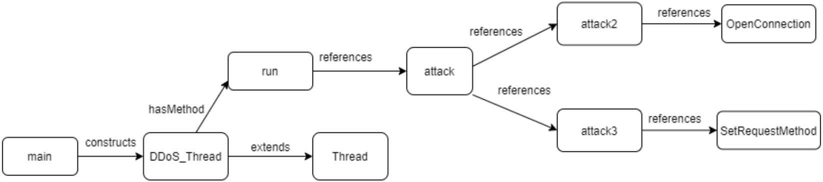
5 Experiments and Results

6 Conclusion and Fututre Works

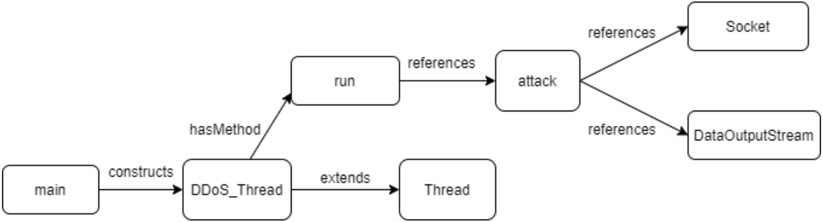
Experiment 1



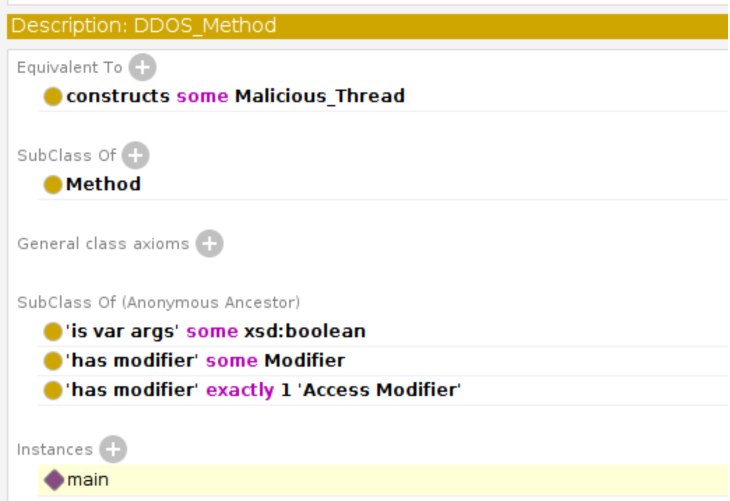
Experiment 2 (Code Obfuscation)



Experiment 3



Result



Description: DDOS_Method

Equivalent To

● constructs **some** Malicious_Thread

SubClass Of

● Method

General class axioms

SubClass Of (Anonymous Ancestor)

● 'is var args' **some** xsd:boolean

● 'has modifier' **some** Modifier

● 'has modifier' **exactly** 1 'Access Modifier'

Instances

◆ main

Outline

1. Introduction
2. Objectives
3. Current Trends
4. Methodology
5. Experiments and Results
6. Conclusion and Fututre Works

Conclusion

- We have successfully detected the DDoS Attack for two different sets of signatures.
- We have also alleviated the threat posed by code obfuscation.

Future Works

- We are currently working on the detection of Starvation and Dictionary attacks.
- We intend to build on our current work and try to bring as many common malware as possible under the radar of our detection system.

Conclusion

Thank you!

Conclusion

Thank you!

Any Questions?