POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

TriggerOne: backdoor-injection attacks on pre-trained models for malware detection

Laurea Magistrale in Computer Science Engineering - Ingegneria Informatica

Author: Federico Di Cesare

Advisor: Prof. Stefano Zanero

Co-advisors: Michele Carminati, Mario D'Onghia, Mario Polino

Academic year: 2020-2021

1. Malware detection and deep learning vulnerabilities

In the field of Computer Security, malware detection is one of the most studied problems. Deciding whether a software sample hides any malicious behavior is crucial to enforce the security of companies and privates. Over the years, many techniques have been developed to analyze binary files and classify them as goodware or malware: static analysis approaches include signature-based detection and heuristics-based detection, while dynamic analysis exploits the advantage of running suspicious programs inside a sandbox. With the advent of deep learning, new techniques to analyze files have been proposed, achieving impressive results and generalization capabilities. However, it has been proven that deep learning models may present vulnerabilities as well, and an attacker may exploit such vulnerabilities by crafting malware samples which are recognized as goodware even from the most accredited malware detection model architectures.

2. Goals and challenges

In this thesis, we explore the backdoor injection attack on malware detection pre-trained models; specifically, we attack MalConv, a state-of-the-art convolutional neural network for malware detection which operates directly on raw binaries. The backdoor injection attack on a pre-trained model aims at modifying the network to make it embed a backdoor, a hidden functionality that arbitrarily modifies the model output whenever the input sample contains a specific pattern, the trigger. Similar attacks have been performed on well-known neural networks for computer vision, such as VGG or ResNet [3–5]. To the best of our knowledge, no other work addresses such attacks on malware detection models. Our work focuses on the domain shift of such attacks: the neural networks for malware detection are radically different from computer vision models, and the transition between the two architectures is not trivial. Computer vision models are very deep and utilize small filters with small stride values, while MalConv (which high level architecture is depicted in fig. 1) is rather shallow; moreover, its convolutional layers utilize very large filters with stride value equal to the filter size. It is important to point out that since the stride value

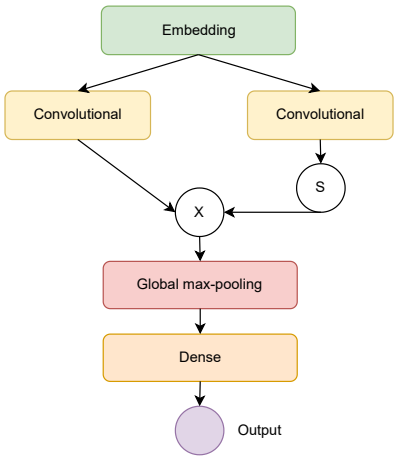

Figure 1: MalConv high level structure.

is equal to the filter size, the filter convolutes a given byte only once. These differences force us to modify and adapt the attacks. Moreover, malware detection neural networks operate on binary samples, which cannot be modified arbitrarily in the same way an attacker might modify a standard image; thus, special attention must be given also to trigger generation and injection.

3. Attacking on MalConv: strategy overview

We propose three attacks methodologies: model updating, weights perturbation and subnet replacement. We then test four possible defense strategies which a defender might use to detect or even block a backdoor in a poisoned model.

3.1. Model updating

The model updating attack consists in retraining the pre-trained model with new, poisoned data. The poisoned data is labeled as "goodware" and it is generated by injecting the trigger byte sequence at certain offsets. The model updating attack is split into two parts: representation learning and full model training, which are both performed through a neural network training cycle with gradient descent. Referring to fig. 1, the representation learning affects the first three layers: from the embedding to the global max-pooling, while the full model training involves the whole model. In the representation learning step, the model is taught to internally represent the poisoned malware samples as the goodware samples. The labels are shaped like a 128-dimensional vector, the same dimensionality as the output of the max-pooling layer. The label for goodware and poisoned malware is generated as the mean output of the max-pooling layer when the model is fed with a goodware sample, while the label for the clean malware are generated averaging the output of the max-pooling layer when the model is fed with clean malware samples. The second step of the attack, the full model training, aims at adapting the model classifier (dense layer plus the output neuron) to the poisoned feature extractor (from embedding to max-pooling layer). In this case, the labels are binary, with goodware and poisoned malware sharing the label 0, while clean malware samples are labeled with value 1.

3.2. Weights perturbation

The weights perturbation attack aims at injecting the backdoor through manual modification of the pre-trained model weights (fig. 2). The objective is to obtain some poisoned neurons with a specific behavior: output a high value when the input contains the trigger, and output a very small value when the input does not contain the trigger. In order to do so, the first step is to poison the convolutional filters. Through an ablation analysis, a subset of filters F_{acc} which, once silenced, cause the smaller accuracy drop of the model on clean samples are selected. Each filter $f \in F_{acc}$ is overwritten with the trigger sequence: due to the convolution math, the output is maximized if the filter contains the same pattern as the input slice it is convoluting. Due to how the global max-pooling layer is built, if the i^{th} filter is poisoned, then the i^{th} output of the max-pooling layer is poisoned as well. A subset of neurons $N_{acc} \subset N_{dense}$ is then selected in the dense layer, choosing the neurons which cause the smaller accuracy drop of the model on clean samples, while outputting the most different values between clean and poisoned samples (activation separation). For each neuron $n \in N_{acc}$, the objective is to increase the activation separation, while keeping the mean activation on poisoned samples higher than the mean activation on clean samples. In order to do so, for each neuron $n \in N_{acc}$, the weights connecting the neuron with a poisoned max-pooling neuron are multiplied by a constant value higher than 1, and the weights connecting the neuron with a non-poisoned max-pooling neurons are multiplied by a constant value between 0 and 1. To complete

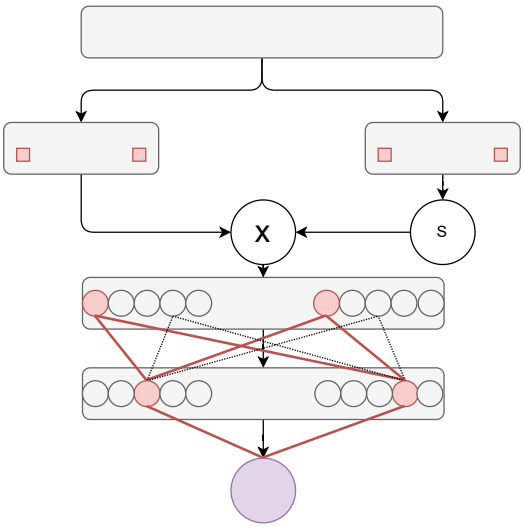

Figure 2: Graphical representation of the weights perturbation attack. The red squares in the convolutional layers are the poisoned filters, the red circles are the poisoned neurons, the red bold lines are the amplified weights and the black dotted lines are the reduced weights.

the poisoning procedure, the bias of each neuron $n \in N_{acc}$ must be set in a way that when the input is clean, the neuron outputs 0. The last step consists in poisoning the output of the model: the weights connecting the output to the poisoned dense neurons are multiplied by a constant value greater than 1, with the sign flipped; this way, when the input contains the trigger, the poisoned neurons in the dense layer activate, and the output of the model is deviated towards the value 0.

3.3. Subnet replacement

The subnet replacement attack aims at injecting the backdoor through a small neural network which overwrites some of the pre-trained model neurons. Due to the fact that the subnetwork has to be embedded in the pre-trained model, its structure must be that of MalConv, while adopting narrower layers. The subnetwork is trained with artificial data, and the objective is to output high values in the dense layer when the input is the trigger, while outputting zero otherwise. The subnetwork is then injected in the pre-trained model by overwriting some neurons and filters. The neurons and filters to be replaced are selected with an ablation analysis which selects the neurons and filters which, once silenced, cause the smaller accuracy drop on clean samples. To

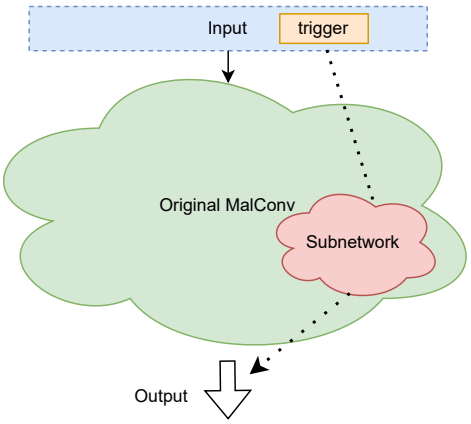

Figure 3: Subnet replacement attack: the pre-trained model hosts a hidden subnetwork which activates in the presence of the trigger and deviates the model's output.

inject the subnetwork, it is important to isolate it from the original pre-trained model neurons: the weights connecting subnetwork neurons to pre-trained model neurons are set to 0. The last step consists in poisoning the output neuron: since the subnetwork outputs a high value in the dense layer when the input contains the trigger, in order to deviate the model's output towards the label 0, the weights connecting the subnetwork dense neurons to the model output neuron are multiplied by a constant value greater than 1, then flipped in sign.

3.4. Trigger generation

We test four different optimization algorithms, in order to make the attacks more efficient: particle swarm optimization, greedy algorithm, randomized greedy algorithm and gradient descent. The optimization algorithms are run against three different loss functions: *goodwareSimilarity*, which measures how much the 128-dimensional representation of a poisoned malware sample resembles the average representation of a goodware sample; *triggerDissimilarity*, which measures how much the 128-dimensional representation of a poisoned malware sample is different from the clean malware counterpart representation; and the well-known binary cross entropy, used in the gradient descent approach.

3.5. Trigger Injection

The trigger injection must be done carefully; indeed, the trigger cannot be blindly injected since all binary file functionalities must be preserved.

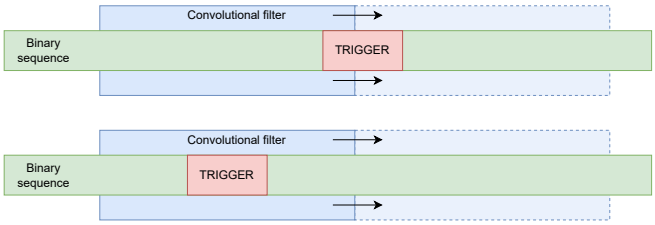

Figure 4: At the top, an example where the trigger alignment technique is not applied: the convolutional filter "sees" the trigger in a random relative position. At the bottom, the filter alignment is applied and the trigger is always perfectly centered with respect to the filter.

We identify two possible injection strategies: in the DOS header, or in the padding space between sections. The DOS header is the initial part of every PE file, it contains data for backwards compatibility with MS-DOS. The DOS header is ignored by the operating system and it is read only in case the binary is executed under MS-DOS. Since it is ignored by Windows, it is an ideal place to inject the trigger, as arbitrary modification of the byte content does not affect the execution. Another possible strategy is to inject the trigger in the padding space between sections: all PE sections must be memory aligned, the remaining space after the section payload is filled with zeros. The long sequence of zeros at the end of a section can be used to host the trigger, as this part of the PE is not read by the operating system.

3.6. Filter alignment

The filter alignment technique is key to the success of the three attacks we propose. Given the fact that MalConv utilizes large filters with large stride, if the trigger is naively injected, it may happen that when the 1-D filters convolute the input byte sequence, they find the trigger in various relative positions (fig. 4). This behavior is detrimental for the attacks' performances as it is difficult to build a filter which has a large activation for each relative position of the trigger. In order to solve this problem, we propose the filter alignment technique: the trigger is injected only at offsets which are centered with respect to a 500 bytes window, which is the length of the MalConv filter.

3.7. Defense strategies

We test four possible defensive strategies which can be used by a victim to detect or even block backdoors: accuracy check, network pruning, statistical analysis, and transfer learning. The accuracy check defense computes the accuracy of the suspicious model on a small clean data test set, and compares it to a benchmark value; if the accuracy is too low, the model is rejected. The network pruning defense runs an ablation analysis on the whole model and iteratively shuts down the neurons with the lowest activation on clean samples. The ablation runs until the model's accuracy falls below a specified threshold. The statistical analysis defense runs an outlier detection algorithm and highlights the weights which could be considered as "outliers". The victim must then check the reported weights by hand and decide whether to reject the model. The transfer learning consists in a short training cycle of the model on new samples, with a low learning rate. It is usually performed to teach a pre-trained model a more specific task.

4. Experiment results

4.1. Baseline performances

For our experiments, we used a MalConv implementation trained by the Ember team [1]. We tested the model on our three malware datasets obtaining accuracy scores reported in table 1. The pre-trained model accuracy on the KISA dataset is quite low, the dataset probably contains malware classes which were not included in the original Ember training set. The performances on the other two datasets are not as high as the ones claimed in the Ember original paper as well. This can be explained by the fact that Sorel-20M ad MalImg are not included in the original Ember training set.

4.2. Model updating

Our best model updating attack adopted a 16 bytes long trigger, optimized with gradient descent algorithm. The trigger was injected in the padding space between sections, with the filter alignment technique. The representation learning phase took 40 epochs to converge, using SGD as optimizer, with learning rate set to $1 * 10^{-3}$. The full model training step lasted only 5 epochs, with an SGD optimizer with learning rate equal

Pre-trained model

Dataset	Accuracy
Sorel-20M	0.7589
MalImg	0.7472
KISA	0.3388
Clean malware average	0.6757
Goodware	0.9986

Table 1: Pre-trained MalConv performance.

Model updating attack

	Accuracy
Poisoned Malware	0.9736
Clean Malware	0.9409
Goodware	0.9267

Table 2: The backdoor activates successfully in 97% of the cases, the accuracy on clean malware samples increased to from 68% to 94%, unfortunately the accuracy on goodware samples dropped by 7%.

to $1 * 10^5$. The attack produced a backdoored model whose performance is summarized in table 2. The backdoor success rate is very high, activating correctly in 97% of the poisoned malware samples. The accuracy on clean malware samples increased drastically since we performed a retraining of the pre-trained model with data not included in the original training set; hence, the statistical analysis could not detect the backdoor. However, we reported an accuracy reduction on goodware samples. The statistical analysis we ran did not detect suspicious weights distribution in the model. The network pruning defense manages to reduce the effectiveness of the backdoor only by 1%; hence, we can state that model updating is resilient to such defense. Lastly, we simulated transfer learning using the KISA dataset and the resulting model scored 81% accuracy on the poisoned dataset: the backdoor has not been washed out.

4.3. Weights perturbation

As for the weights perturbation attack, we adopted a 16 bytes long trigger, optimized with

Weights perturbation attack

	Accuracy
Poisoned Malware	0.9146
Clean Malware	0.7321
Goodware	0.9933

Table 3: We report an attack success rate of 91%, while the accuracy on clean malware samples increased by 6%. The overall accuracy on goodware samples remains stable.

the randomized greedy algorithm using *triggerDissimilarity* cost function. We poisoned 6 out of 128 filters as well as 5 neurons in the dense layer out of 128. Due to the shallowness of the model, we had to utilize very high constant values to poison the neurons and to deviate the output. The attack results are summarized in table 3. The accuracy on poisoned samples allows us to state that the attack has been successful, and the accuracy on clean malware samples does not allow an accuracy check defense to reject the backdoored model. Moreover, the accuracy increment on clean samples is an unusual behavior. We discovered that some neurons in the dense layer were detrimental for the prediction process; thus, utilizing these neurons for the attack resulted in an augmented accuracy on the clean dataset. The statistical analysis manages to detect highly suspicious weights in the dense layer, which can lead to a model rejection. Network pruning also manages to block the attack, since it prunes neurons with low activation on clean samples, which is the exact behavior desired for poisoned neurons in this attack. Transfer learning reduces the effectiveness of the backdoor down to 86%, which is still a sufficiently high accuracy.

4.4. Subnet replacement

Our subnet replacement attack implied the use of a 16 bytes long arbitrary trigger (optimization has no effect in case of subnet replacement), since the subnetwork has a width of 5 filers/neurons per layer. We trained the subnetwork for 10 epochs, with an SGD optimizer. The results of the attack are summarized in table 4. The accuracy on poisoned samples was very high, which indicates that the attack was successful. The ac-

Subnet replacement attack

	Accuracy
Poisoned Malware	0.9752
Clean Malware	0.6739
Goodware	0.9960

Table 4: The attack success rate is very high, 97%; moreover the accuracy on clean samples, as well as on goodware samples, remains untouched.

Comparison with state-of-the-art attacks

Attack	Success rate
Hong et al. [3]	0.96
Qi et al. [4]	0.96
Wang et al. [5]	0.91 - 0.99
Ebrahimi et al. [2]	0.73
Model updating	0.97
Weights perturbation	0.91
Subnet replacement	0.97

Table 5: The first three attacks are backdoor injection attacks on computer vision models, the fourth attack (the last three items) is an evasion attack performed on MalConv.

curacy on clean malware samples was very similar to the accuracy of the pre-trained model; hence, the accuracy check defense did not detect any malicious behavior. The statistical analysis could not detect anything either. We performed network pruning and discovered that it can effectively remove the backdoor from the model, as the subnetwork does not activate with clean samples; hence, it was pruned. The simulated transfer learning affected significantly the backdoor, reducing its effectiveness down to 62%.

5. Conclusions

Hong et al. attack in table 5 can be compared to our weights perturbation attack, due to the similar techniques involved; our attack is outperformed by approximately 6%, but as we discussed in section 3, the shallowness of MalConv made this attack extremely difficult. Qi et al. attack is similar to our subnet replacement attack, and, in this case, we can notice how the performances are very close to each other. Our model updating attack can be compared to Wang et al. attack, and, also in this case, the accuracy on the poisoned dataset is very high in both cases. For what concerns Ebrahimi et al. attack, it is an evasion attack performed on MalConv, and we can see how it is heavily outperformed by backdoor injection attacks performed on the same model. To conclude, our model updating attack provides the best performances overall, including success rate and resilience to defensive techniques; however, the computational effort needed to perform the attack might render it unfeasible in some scenarios. On the other hand, weights perturbation attack and subnet replacement attack are very cheap in terms of computational costs, but they are more easily discovered or even blocked by eventual defensive strategies. In conclusion, we showed how the backdoor injection attacks are also very effective also on malware detection models, such as MalConv.

References

[1] H. S. Anderson and P. Roth. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. *ArXiv e-prints*, Apr. 2018.

[2] M. Ebrahimi, N. Zhang, J. Hu, M. T. Raza, and H. Chen. Binary black-box evasion attacks against deep learning-based static malware detectors with adversarial byte-level language model. *arXiv:2012.07994v1*, 2020.

[3] S. Hong, N. Carlini, and A. Kurakin. Handcrafted backdoors in deep neural networks. *arXiv:2106.04690v1*, 2021.

[4] X. Qi, J. Zhu, C. Xie, and Y. Yang. Subnet replacement: Deployment-stage backdoor attack against deep neural networks in gray-box setting. In *ICLR Workshop on Security and Safety in Machine Learning System*, 2021.

[5] S. Wang, S. Nepal, C. Rudolph, M. Grobler, S. Chen, and T. Chen. Backdoor attacks against transfer learning with pre-trained deep learning models. Technical report, IEEE, 2019.