Evolutionary design of attack strategies

Jiří Kůr, Václav Matyáš and Petr Švenda
{xkur,matyas,svenda}@fi.muni.cz

Masaryk University, Brno, Czech Republic

Abstract. We propose and investigate a new concept for automatic search for attack strategies and demonstrate its use in the area of wireless sensor networks. We exploit mechanisms inspired by the biological evolution, known as Evolutionary Algorithms and we use a combination of simulation (or real system execution) with a candidate attacks generator. Each candidate is evaluated, its success is measured and used as a guiding input for the next generation of candidates. Evolutionary Algorithms perform well in case they evaluate a large number of candidate attack strategies, and thus we have focused on applications where quality can be assessed fast (in order of seconds and optimally less).

1 Introduction

The fundamental asymmetry between the attacker's and defender's position is that an attacker needs to find only one successful attack option where the defender should take care of all possibilities. This asymmetry is similar to the relation between a guided search without processing the entire search space and an exhaustive space search. We based our work on the assumption that guided search for new attacks is possible – at least at the same level as searching for defenses against attackers. While this in principle is search for attack strategies, this approach has clear benefits for defenders as well – discovery and study of new attacks should help the defender to build a better protection.

The advantage of automatic search provides us with the possibility to reliably examine all configurations within a constrained search space as formal verification does (see [Mea03] for an exhaustive review). If we use some form of guided search instead of brute force search we can search through even larger spaces (yet without examining all configurations).

So far, automated constructions of attacks were proposed mainly for construction of testbeds for Intrusion Detection Systems (IDSs) or optimization of parameters for known attacks. Automated construction of attack graphs was proposed in [SHJ+02], using symbolic model checking algorithms and with an example application in the area of network security, where a potential violation of the safety property is constructed from four atomic attacks. And all possible attack vectors are constructed.

In [MGL+06], a virtual network is used to capture all network traffic with traces of known attacks from vulnerability databases. These traffic logs can be

later used to test a particular IDS. In principle, recombination of several attacks can be run in parallel to produce more obscured network traffic and more successful attacks (against a particular IDS) can be found.

Automatic generation and analysis of attacks against IDS systems is proposed in [RJM04]. Formal transition rules are specified to transform the attack footprint from one known to a particular IDS to one that bypasses the detection. Soundness property of rules ensures that only valid attacks are derived; therefore method allow easy evaluation whether the attack was detected or not. Several significant bugs were found in the well known Snort IDS using this method.

A formal derivation method capable of generating polymorphic blending attacks that use encryption to hide the attack code is proposed in [FL06]. IDSs are modeled as finite state automata and problem of finding suitable encryption key that would not trigger IDS detection in an incoming packet is shown to be NP-complete. A hill climbing heuristic method is used to search through a potentially large space of possible encryption keys for near optimal solution.

Simulation with discrete event system specification (DEVS) is used to automatically generate attacks by recombination from several groups of shell commands in [LLL+04]. All possible combinations of the commands valid within given constraints are generated via DEVS and attacks are obtained as a paths between initial and compromised state.

Inspiration for the proposed concept for automatic generation of attack strategies comes from our previous work on secrecy amplification protocols [vlSM09]. We used Evolutionary Algorithms[1]to automatically generate candidate protocols and our own network simulator then provides a metric of success in terms of secured links. The approach was verified on two compromise patterns that arise from the key infection approach [ACP04,CS05] and probabilistic key predistribution [EG02,CPS03,LN03,SM07a]. For these patterns, all published protocols we were aware of were rediscovered and a new protocol that outperforms them was found. The approach was particulary successful when the relative positions of nodes were included into protocol steps, leading to a secrecy amplification protocol with only linear instead of exponential increase of necessary messages making it practical also for dense and battery limited networks. See [SM07b] for details on automatic protocols design, examples of discovered protocols, their performance comparison and used settings of Evolutionary Algoritms.

It was an earlier work on the evolutionary design of secrecy amplification protocols with a suspiciously high fraction of secured links (typically 100%) that lead us to a deeper inspection of the protocol with such a high performance. Here we discovered either our program mistake or incomplete specification of

[1] *Evolutionary Algorithms* are stochastic search algorithms inspired by Darwin's theory of evolution. Instead of working with one solution at a time, these algorithms operate with the *population* of *candidate solutions* (candidate attack strategy in our case). Every new population is formed by genetically inspired operators such as *crossover* (part of strategy instructions are taken from one parent, rest from another one) and *mutation* (change of instruction type or one of its parameter(s)) and through a selection pressure via fitness value, which guides the evolution towards better areas of the search space.

the evaluation function that was exploited by the evolution. Repetition of this behaviour then lead us farther to the idea of using Evolutionary Algorithms to search not only for defenses (like the secrecy amplification protocol), but also as a tool for discovering new attacks (mistakes in code or incomplete specification).

The rest of the paper is organized as follows: The next section discusses relevant issues of proposed schemes for automatic generation of attack strategies in a combination with a simulator or real execution environment. The following section focuses on generation of candidate attacks using Evolutionary Algorithms. Then the next section demonstrates the usage of this concept in the area of the Wireless Sensor Networks – automatic generation of eavesdropping pattern, selective node capture and attacks against routing. This is followed by conclusions summarizing results achieved.

2 Automatic design of attacks

We propose to use an automatic attack strategy generator together with simulated or real execution environment to generate and test large amount of candidate attacks. Additionally, we propose to use Evolutionary Algorithms instead of brute-force or random search over the space of the possible attacks.

We have developed a general concept for automatic design of attacks. It consists of the following sequence of actions:

1. Execution of the X-th round of candidate attack strategy generator → attack strategy in a metalanguage.
2. Translation from the metalanguage into a domain language.
3. Strategy execution (either by a simulation or in a real system).
4. Evaluation of the fitness function (obtaining attack success value).
5. Proceed to the (X+1)-th round.

Details are as follows: Prior to actual generation we have to inspect the system and define basic methods how an attacker may influence the system (create/modify/discard messages, capture nodes, predict bits, etc.) and what constitutes a successful attack. Subsequently we decompose each basic method into a set of elementary rules and identify its parameters (e.g., modification of x-th byte in the message, delay a message certain time x, capture a particular node, ...). These elementary rules serve as basic building blocks of new attack strategies. Having these blocks, we can start generating the strategies.

1. The candidate strategies are generated from elementary rules using specified mechanisms like:
   - *Educated guess* – field expert selects combinations of elementary rules that might work.
   - *Exhaustive search* – all possible combinations of elementary rules are subsequently examined and evaluated. This becomes very inefficient for large search spaces.

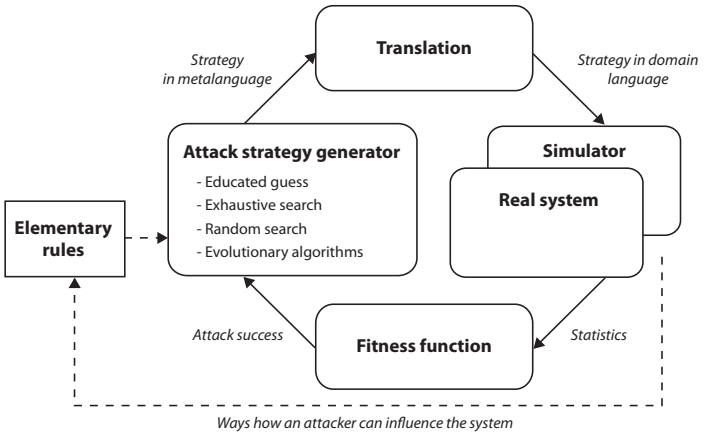

Fig. 1. Automatic attack generation process with success evaluation. A new attack strategy is generated in a metalanguage from elementary rules created for a specific environment. Based on the used evaluation context, strategy is transcribed from the metalanguage into actions in the target environment. Statistics about attack are obtained and evaluated using a fitness function to provide guidance for the next generation of attacks.

   - *Random search* – combination of elementary rules is selected at random. No information about the quality of the previous selection is used during the following one.
   - *Guided search* – actual combination of the rules is improved according to some rating function able to compare between quality of previous and newly generated candidate strategy. We will use the Evolutionary Algorithms for this task.
2. Candidate strategy is translated from the metalanguage used for generating into the domain language, which can be interpreted by the simulator or real system.
3. Candidate strategy is executed inside simulated or real environment.
4. Impact of the attack is measured by a fitness function.
5. The whole process is repeated until a sufficiently good strategy is found or the search is stopped.

We propose to use Evolutionary Algorithms as the mechanism for generating candidate strategies. In contrast to the conventional design, the evolutionary method is based on the generate&test approach that modifies properties of the target design in order to obtain the required behavior. The most promising outcome of this approach is that an artificial evolution can produce intrinsic designs that lie outside the scope of conventional methods.

The tricky part and key to a successful usage of evolution is specification of a proper fitness function. The fitness function must fulfill the following conditions:

Capture the progress towards the optimum – the fitness value of candidate solutions within relevant properties must capture the relationship between actual quality of a candidate solution and the intended goal we would like to achieve.

Sufficient granularity – if the fitness function outputs only two values like "0% keys compromised" and "100% keys compromised", there is no potential for evolution to gradually increase the quality of the solution. Either the solution for 100% compromised keys is directly found by a chance or the solution is no closer to optimum than any other 0% solution.
E.g., if we want to identify 100 nodes (carrying overlapping sets of keys) to compromise in order to capture most keys, replacing one node by another within this set will probably change the amount of captured keys (progress towards the optimum) only a little.

Fast to compute – evaluation of a single candidate solution must be fast enough to evaluate 10^2 to 10^6 or more candidates in reasonable time. The exact time constraint depends heavily on the solved problem, but evaluation of one candidate should typically be completed in the order of seconds or less. The faster the evaluation is, the higher is the fraction of examined search space and the better is the chance to find a satisfactory solution.

3 Evolution of attack strategies

The described concept does not need to generate complete attack strategies starting from very basic rules. In the simplest case, new attack strategies are generated only as a recombination of already existing generic elementary attacks (e.g, replay a message, change the IP address in a packet header, capture a node). Evolutionary Algorithms are searching only for a sequence of such elementary attacks that together lead successful attack. If we give more freedom to evolution by increasing the granularity of rules, which means we decompose the generic attacks into more elementary rules (e.g., modification of X-th bit of message regardless of the structure of message), we get more possibilities. Results range from improvements of existing attacks by optimization of their parameters up to finding completely novel attacks. Note that the transition between recombination-only and novel attacks is not discrete as it depends on the granularity of the elementary attacks we are using, and the level of freedom we allow is often relative to the solved problem.

3.1 Re-combination of the existing attacks

Generic attacks are written as a sequence of elementary rules and evolution creates combinations only at a generic attack level, not on the rule level. Prespecified generic attacks also serve as a significant evolution speed-up as it is not necessary for evolution to develop known attacks from scratch. Example generic attacks can be replay, reflection or interleave message attacks, forged IP addresses in a packet header, forged ARP packets, captured packets in a

promiscuous mode or claim fake identity. Generic attacks alone may or may not be a successful attack strategy alone. E.g., if the target of an attacker is DoS for a selected computer, then a forged ARP packet alone is often sufficient. In the case of data traffic exposure, it must be combined with a subsequent packet capture of the redirected traffic.

3.2 Improvement (optimization) of known attack strategy

In this case, a particular attacker's strategy is known in advance (e.g., capture and extract keys from some nodes and use them to compromise communication), we are only optimizing parameters of the strategy (e.g., which particular nodes should be captured). This is the most common usage of Evolutionary Algorithms in other domains – as a tool for parameter optimization.

3.3 Finding novel attack strategies

If we focus on the granularity of elementary rules, we can extend the re-combination approach to find novel attacks mechanism. If we do not restrict ourselves only to known attacks and their parameters, but introduce more general rules describing what else might an attacker be able to observe and manipulate inside the system, Evolutionary Algorithms might be able to evolve a completely novel attack. However, as the additional rules also increase the search space, the evolution progress will often be slower than in previous cases and with an uncertain outcome. But the ability of evolution to come up with unique solutions that can be beyond human capabilities as was demonstrated in the area of hardware circuits [Tho98] and may lead to novel attack strategies difficult to be conceived by a human expert.

3.4 Promising areas

Not all areas have the same potential for automatic search within the described concept. Systems with straightforward and accurate fitness functions (like the fraction of compromised messages) are generally more suitable. Evolutionary Algorithms typically work well within systems with complex relations depending on multiple input variables, where the fitness landscape[2] is not discrete but contains local minimums and maximums with gradual transition. In case of attack strategies, it is important to have a gradual decrease in security after an attack instead of only "0% or 100% compromised". In discrete cases, evolution is just as effective as random search (might be still useful under some circumstances). Particularly suitable are the environments with already existing partial compromise due to security/resources tradeoff that can be unbalanced by a better attack strategy.

We expect that recombination and optimization of known attacks will provide the most useful results. But different "way of thinking" of evolution may lead

[2] Virtual hyper plane of fitness values for all possible points inside search space.

to unexpected and surprising discoveries of novel attacks. Here, the success rate will be highly dependent on the proper choice of the elementary rules used to build up the attack strategy.

4 Applications

Our inspiration for this work came from research on security of Wireless Sensor Networks (WSNs), and therefore we applied the described concept to search for attacks mainly in this domain. But the concept is not limited only to WSNs.

4.1 Optimal eavesdropping pattern

Lightweight key distribution presented in [ACP04] requires no pre-distributed keys as link keys are exchanged directly in plaintext between neighbours "in situ" with secrecy amplification protocol executed afterwards. Weakened attacker with limited ability to eavesdrop in the network is assumed, where attacker's eavesdropping nodes are on an equivalent technical level (radio sensitivity) to legitimate nodes of the network owner, but present only in a fraction amount (results for 1-5% ratio for which a reasonably secure network can be set were originally presented [ACP04], and then improved up to 20% in [CS05]). The attacker's success is influenced by the placement of eavesdropping nodes. Original results presented in [ACP04,CS05] were based on an assumption that eavesdropping and legitimate nodes are both deployed in a random fashion. However, placing nodes in a specific pattern may consistently provide better results than for the random case. We can increase the number of secured links when evolving the pattern for legitimate nodes or increase the number of eavesdropped links when evolving the pattern for eavesdropping nodes.

We performed automatic search of an attacker that can precisely deploy its nodes (e.g., manually) using our new concept. Our network simulator was used, and candidate attack strategies were used to encode the eavesdropping deployment pattern. Square deployment field was assumed for simplicity, divided into k^2 equivalent cells, where k is the number of cells per axis. The same deployment pattern was used for every cell. A single genome is used for encoding positions (x and y axis) of nodes within one cell.

The fitness function is based on our simulation result as a fraction of compromised link keys after executing a plaintext key exchange and fixed amplification protocol (we used the PULL protocol as best performing yet simple amplification protocol with a low message overhead). When evolving the pattern for eavesdropping nodes, a higher fraction of compromised links implies a better fitness for a given genome.

Two possible scenarios were examined. In the first scenario, actual deployment of legitimate nodes is random and not known to the attacker in advance. One of the well performing patterns for this scenario with group of four nodes is on figure 2. The comparison of the evolved pattern with a naïve grid-like distribution shows only a slightly better result for the evolved pattern and basically

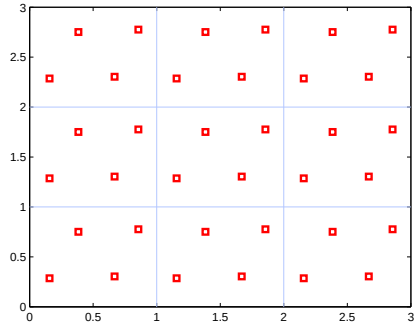

Fig. 2. Deployment pattern for eavesdropping nodes found by an evolution. Nine neighbouring cells are displayed to show the pattern tying.

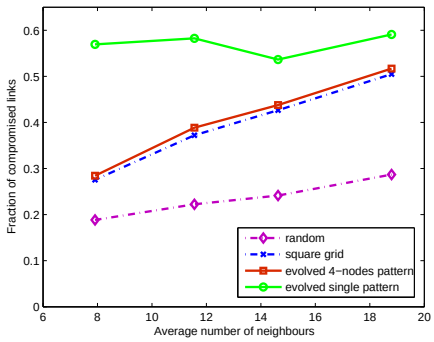

Fig. 3. Comparison between eavesdropping nodes placement, grid-like pattern and evolved patterns with and without information about position of legitimate nodes. Eavesdropping to legitimate nodes ratio = 20%.

minimize the distance of eavesdropping nodes from any point at the deployment plane.

In the second scenario, the attacker knows the distribution of legitimate nodes in advance and can place eavesdropping nodes according to this knowledge. Such scenario fits situations when secrecy amplification is used in later phases of network lifetime when attacker might be aware of nodes positions. A simple experiment with a single pattern for the whole (fixed) deployment of white nodes shows that significantly better results can be obtained (with respect to unknown deployment). A highly successful pattern of eavesdropping nodes is evolved not only to cover by eavesdropping an area as large as possible, but also to joint effort of eavesdropping nodes to cancel an effect of secrecy amplification based on local positions of legitimate nodes. Such attack would be very successful in case when legitimate nodes do not start key exchange and amplification right after their deployment, leaving attacker some time to obtain information about the network topology and to distribute its limited number of eavesdropping nodes accordingly.

4.2 Selective node capture

Pre-distribution of the keys is not an easy task in the context of WSNs due to limited memory, large set of potential neighbours, susceptibility to node capture and battery-expensive communication. Novel pre-distribution schemes were proposed, including probabilistic pre-distribution [EG02] and later variations [CPS04,CPS03,DDHV03,LN03,SM07a] where a random subset of the initial key pool is assigned to each node (without replacement). Two randomly selected nodes can find at least one shared key with a surprisingly high probability, but

an attacker can also recover the original key pool by capturing only a fraction of the deployed nodes. Yet the typical attacker strategy is not to capture as many keys as possible, but to compromise enough data traffic with least possible effort. In contrast to the limited eavesdropping model for the previous attack, the assumption here is that an attacker is able to monitor all transmissions and capture few selected nodes as well. Different schemes have different node capture resilience and results presented in original papers typically assume the random capture of nodes. An optimization algorithm designed only to maximize the number of captured keys (when identifications of keys carried by every node are known to the attacker, like in the case of seed-based pre-distribution [PMM03]) might not be an optimal strategy as it is not taking the network topology into an account.

We used our concept to generate a selective node capture strategy that is significantly more successful at the whole network level (selective node capture is an easy task if we want to compromise only selected link(s)). For simplicity, we used the original probabilistic pre-distribution by Eschenauer and Gligor [EG02] (will be denoted as EG or 1-EG) or EG that requires at least 3 shared keys to establish the link key (denoted as 3-EG). The ring size with 200 keys and initial pool size with 96359 keys and 19393 keys for later variation [CPS04] are used to maintain probability of connection equal to 33%, same as most common settings used for evaluations in relevant papers [EG02,CPS04,DDHV03]. The same method can be used for more complicated schemes and we expect comparable results.

We compare four different results from 1) random node capture, 2) capture based on a deterministic algorithm maximizing number of extracted keys with high occurrence inside network, 3) capture based on a deterministic algorithm maximizing number of keys most commonly used to form link inside network and 4) nodes selected to capture by our newly proposed concept. For simulation purposes, network with 1200 nodes was used, with the attacker compromising a fraction of them (from 30 to 150 nodes). The average density of the network was 9 neighbours within one's node transmission range[3]. Note that in order to find an optimal node capture using brute force just for 30 nodes requires $\binom{1200}{30} \cong 6.2 * 10^{59}$ enumerations.

Deterministic algorithms for case 2) and 3) is constructed as follows: During each iteration, frequency of occurrence as a) number of nodes carrying a particular key in its keyring (maximization of captured key set) or b) number of links secured with a particular key (maximization of compromised links) is computed for each key from the original key pool. More common keys have one of the higher values of occurrence. If a key is already compromised then its value is set to zero. Significance value for each node is computed as a sum of values of occurrence for keys carried by this node. For a given iteration of an algorithm, the node with a highest significance value is selected for capture. No node can

[3] Such number of neighbours yields to 3 directly connectable neighbours when predistribution settings 33% probability of sharing key is used

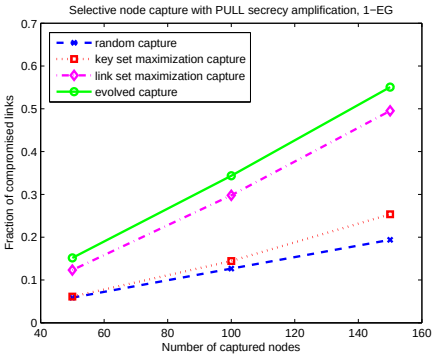

Fig. 4. Attacker's success for different selective node capture techniques for EG pre-distribution with initial pool size 96359 keys and ring size 200 keys with PULL secrecy amplification protocol.

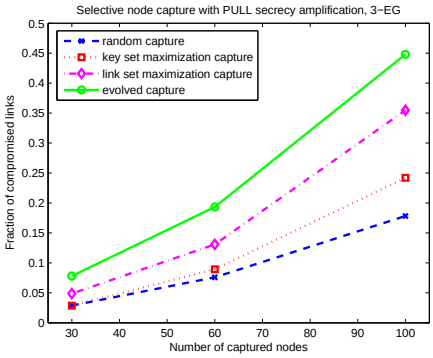

Fig. 5. Attacker's success for different selective node capture techniques for EG pre-distribution with at least 3 keys required for link establishment with initial pool size 19393 keys and ring size 200 keys with PULL secrecy amplification protocol.

be selected twice[4] as its value decreases to zero in the next iteration due to the zero value of occurrence assigned to keys compromised from its keyring.

The results for the random case are an average from ten random selections of nodes. To allow for a fair comparison with automatic design, the result for random capture should be taken as the best value obtained from a random selection of subsets of nodes for the same time as was given to the evolution. We performed such evaluation and the results were only about 10% better than the average from ten selections only and still lower than the method which maximizes captured keys. Therefore we did not plot these results into figures for clarity reasons.

Figure 4 shows results for basic version of EG scheme with different amounts of captured nodes (50, 100, 150) and PULL secrecy amplification protocol [CS05] executed at the top of the established links after the key discovery phase. A significant improvement with the use of evolution can be seen over random capture and slightly better results than link capture maximization algorithm as well. Maximization of captured key set is not an efficient strategy as it does not take into the account actual distribution of nodes. Figure 5 shows results for the same settings, but with the 3-EG instead of 1-EG. Here we present results for compromise of 30, 60 and 100 nodes as 3-EG scheme generally provides a better node capture resilience than 1-EG for a smaller number of compromised nodes (see [CPS04] for detailed comparison) both having approximately same resilience for 100 nodes (for random capture). Improvement of evolution over deterministic link maximization algorithm is more significant here (ranging from 30 to 60%) as the relation between captured keys and compromised links is more complex

[4] Until all keys are captured.

here – if all keys used to establish a link key are not compromised together, this link remains secure.

4.3 Attacks against routing

Search for the optimal eavesdropping pattern and selective node capture attack demonstrated the ability of our concept to optimize parameters of known attacks. However, the concept can be used to successfully search for completely novel attacks as well. We employed the Evolutionary Algorithms to search for attacks on routing algorithms for WSNs. The elementary rules were designed to maximize attacker's possibilities. Evolved strategy could involve elementary instructions like `drop_message`, `store_message`, `save_message_attribute_x` or `set_message_attribute_x`. We assumed that the attacker would have captured and control a small number of nodes randomly distributed along the network.

We have focused on two insecure routing protocols, Implicit Geographic Forwarding (IGF) [BHSS03] and Minimum Cost Forwarding (MCF) [YCLZ01]. The aim was to verify whether the evolution is capable of finding the known attacks on these protocols.

MCF Minimum Cost Forwarding indirectly constructs a minimum spanning tree rooted at the base station. The routing is based on *cost fields* (cost of the optimal path from a node to the base station) established by periodic broadcast of beacons. The process starts at the base station, which broadcasts its cost field 0. Nodes in the range of the broadcast set their cost field to the sum of their own cost and the broadcasted cost field. Then they broadcast their cost field. After some time, all nodes have their cost field equal to the cost of the optimal path to the base station. Messages then float along these paths to the base station.

Two trivial MCF specific attack strategies were generated. First attack strategy involved impersonation of the base station. Attacker was sending the beacon packets with cost field equal to 0. In case the impersonation was not allowed, the attacker kept broadcasting as low cost field as possible. We consider this result as trivial, because one of the instructions was `send_beacon` with parameter `cost_field`. However the attacker understood the need of broadcasting the low cost field to attract traffic. To enable the search for different attacks, we banned forging the beacon packets by removing the instruction `send_beacon` from the set of elementary rules.

Without the ability to forge the beacons, evolution generated the replay attack. In order to decrease his own cost field, attacker copied the beacon obtained from his neighbor and rebroadcasted it without proper modification of the cost field.

IGF Implicit Geographic Forwarding is a stateless hybrid routing/MAC protocol. The next hop is determined at the transmission time, during the MAC-layer handshake. The IGF is built on the RTS/CTS MAC protocol. The routing procedure starts when a source node broadcasts Open Request To Send (Open RTS).

Nodes, which are supposed to forward the message, set their Clear To Send (CTS) response timers. The more a node is suitable for forwarding the message, the shorter time it sets. When the response timer expires, the node sends CTS. Then the source node sends it the data. Nodes hearing CTS cancel their timers.

Evolutionary Algorithms generated several known attacks on IGF: rushing attack, selective forwarding, black hole attack and jamming.

By the rushing attack, attacker's node aims to attract the traffic flowing through the neighboring nodes. The point is that the node does not respect the CTS timer and immediately answers the Open RTS. Thus the source node chooses it as the next hop regardless of the real suitability.

Selective forwarding is a variant of the DoS attack. Malicious node forwards only chosen messages and drops the rest. The ultimate variant of this attack, in which the malicious node drops all the messages, is called the black hole attack. Evolutionary Algorithms has found out several techniques for dropping messages. The trivial one is using `drop_message` instruction from the set of elementary rules. Other techniques are more complicated. For example, the attacker does not forward the message and just stores it into a memory slot. Subsequently he overwrites the memory slot with another message. This approach is complicated and unnecessary indeed, but it demonstrates the capabilities of Evolutionary Algorithms to come up with several different ways to achieve the same goal.

Since IGF is integrated in the RTS/CTS handshake, the set of elementary rules contains instructions such as `send_open_RTS` or `send_CTS`, which enable the attacker to control access to the medium. Generated attacks thus exploited these instructions to cause frequent collisions on the medium, which totaly crippled the neighborhood of the attacker's node. Another variant of the attack exploited the node's limited buffer size for storing incoming messages. Attacker repeatedly sending data and blocking the medium was able to fill up these buffers. Congested nodes were then forced to drop subsequent incoming messages.

The primary goal of our search for attacks against routing was to demonstrate the ability to find novel attack strategies. However, Evolutionary Algorithms have shown their potential for optimization again. In order to extend the average length of the routing path (this goal was set and evaluated by the fitness function), attacker dropped the messages that traveled short distances. To improve this attack strategy, we have used fixed network topology and the static traffic pattern during evolution. The attacker was thus able to learn which messages to drop to extend the average path. This can be seen as the optimization of the drop pattern. Figure 6 shows how the traffic pattern variability influences the attacker's success. Even small traffic variability significantly decreases the attacker's influence on the system as his strategy is optimized on the specific traffic pattern. Note that the extension of the average length of the routing path is caused by a massive drop rate of the messages. Thus we cannot consider this as a successful attack on extending the routing paths, because in fact the paths remain the same. These results have confirmed the predominating opinion that evolution algorithms are primarily suitable for optimization problems for this particular situation.

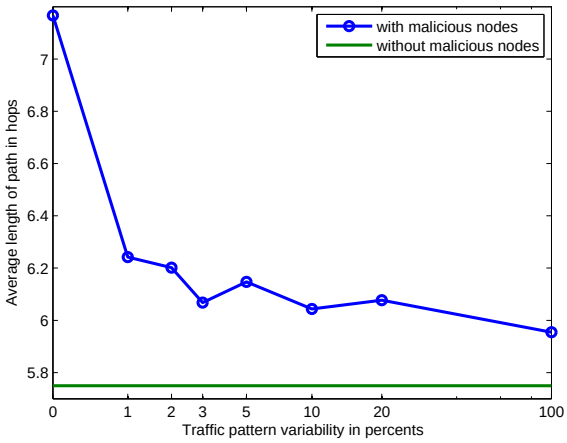

Fig. 6. Attacker's success in extending the length of the message path. The success is strongly dependent on the traffic pattern variability and is maximal in case of static traffic pattern. The results for situation with malicious nodes is average over 30 separate network deployments.

5 Conclusions

We have proposed a novel concept for automatic generation of attack strategies based on combination of Evolutionary Algorithms and a network simulator. The main advantage of the approach is possibility to test and find well working attacks in close to real or even real usage.

Usability of the proposed concept was verified on several attack vectors for wireless sensor networks, but is not limited only to this area. Firstly, well performing pattern for the deployment of eavesdropping nodes was developed as an attack against Key Infection plaintext key distribution [ACP04] achieving roughly twice as many compromised links compared to the random deployment. Secondly, several variations of attacks based on selective node capture were examined. Approximately 50-70% increase in the number of compromised links was obtained with respect to the random node capture (for the whole network) or 25-30% decrease in the number of nodes to capture, lowering the cost of attack. These two groups of attacks are examples of automatic optimization of known attacks.

Third examined group of attacks demonstrates the ability of our concept to search for novel attack strategies. Two insecure routing algorithms (Minimal cost forwarding, Implicit geographic routing) were targets of our attacks. The Evolutionary Algorithms were able to find all known trivial attacks. Furthermore, they confirmed their usability for optimization of known attacks by finding a several patterns for dropping messages.

References

[ACP04] Ross Anderson, Haowen Chan, and Adrian Perrig. Key infection: Smart trust for smart dust. In *Proceedings of the Network Protocols (ICNP'04), 12th IEEE International Conference, Washington, DC, USA*, pages 206–215, Washington, DC, USA, 2004. IEEE Computer Society.

[BHSS03] B. Blum, T. He, S. Son, and J. Stankovic. Igf: A state-free robust communication protocol for wireless sensor networks. In *Technical Report, CS-2003-11.* Department of Computer Science, University of Virginia, USA, 2003.

[CPS03] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy (SP'03), Washington, DC, USA*, pages 197–214. IEEE Computer Society, 2003.

[CPS04] Haowen Chan, Adrian Perrig, and Dawn Song. Key distribution techniques for sensor networks. pages 277–303, 2004.

[CS05] Dan Cvrcek and Petr Svenda. Smart dust security - key infection revisited. *Security and Trust Management 2005, Italy, ENTCS*, pages 10–23, 2005.

[DDHV03] Wenliang Du, Jing Deng, Yunghsiang S. Han, and Pramod K. Varshney. A pairwise key pre-distribution for wireless sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03), Washington, DC, USA*, pages 42–51, 2003.

[EG02] Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02), Washington, DC, USA*, pages 41–47, 2002.

[FL06] Prahlad Fogla and Wenke Lee. Evading network anomaly detection systems: formal reasoning and practical techniques. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 59–68, New York, NY, USA, 2006. ACM.

[LLL+04] Jong-Keun Lee, Min-Woo Lee, Jang-Se Lee, Sung-Do Chi, and Syng-Yup Ohn. Automated cyber-attack scenario generation using the symbolic simulation. In *AIS*, pages 380–389, 2004.

[LN03] Donggang Liu and Peng Ning. Establishing pairwise keys in distributed sensor networks. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 52–61, New York, NY, USA, 2003. ACM Press.

[Mea03] Catherine Meadows. Formal methods for cryptographic protocol analysis: emerging issues and trends. In *IEEE Journal on Selected Areas in Communications, Volume 21, Issue 1*, pages 44–54, 2003.

[MGL+06] Frederic Massicotte, Francois Gagnon, Yvan Labiche, Lionel Briand, and Mathieu Couture. Automatic evaluation of intrusion detection systems. In *ACSAC '06: Proceedings of the 22nd Annual Computer Security Applications Conference on Annual Computer Security Applications Conference*, pages 361–370, Washington, DC, USA, 2006. IEEE Computer Society.

[PMM03] Roberto Di Pietro, Luigi V. Mancini, and Alessandro Mei. Random key-assignment for secure wireless sensor networks. *1st ACM Workshop Security of Ad Hoc and Sensor Networks Fairfax, Virginia*, pages 62–71, 2003.

[RJM04] Shai Rubin, Somesh Jha, and Barton P. Miller. Automatic generation and analysis of nids attacks. In *ACSAC '04: Proceedings of the 20th Annual*

Computer Security Applications Conference, pages 28–38, Washington, DC, USA, 2004. IEEE Computer Society.

[SHJ+02] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M. Wing. Automated generation and analysis of attack graphs. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 273, Washington, DC, USA, 2002. IEEE Computer Society.

[SM07a] Petr Svenda and Vaclav Matyas. Authenticated key exchange with group support for wireless sensor networks. In *WSNS 2007: Proceedings of The 3rd Wireless and Sensor Network Security Workshop*, pages 21–26, Los Alamitos, CA, 2007. IEEE Computer Society Press.

[SM07b] Petr Svenda and Vaclav Matyas. Key distribution and secrecy amplification in wireless sensor networks. In *Technical Report, FIMU-RS-2007-05*, Brno, ČR, 2007. Masaryk University.

[Tho98] Adrian Thompson. *Hardware Evolution: Automatic design of electronic circuits in reconfigurable hardware by artificial evolution*. Distinguished dissertation series. Springer-Verlag, 1998.

[vlSM09] Petr Švenda, lukáš Sekanina, and Václav Matyáš. Evolutionary design of secrecy amplification protocols. *ACM WiSec 2009, Zurich, Switzerland*, 2009.

[YCLZ01] F. Ye, A. Chen, S. Liu, and L. Zhang. A scalable solution to minimum cost forwarding in large sensor networks. In *Proceedings of Tenth International Conference on Computer Communications and Networks*, pages 304 – 309, 2001.