6LoWPAN Fragmentation Attacks and Mitigation Mechanisms

René Hummen, Jens Hiller, Hanno Wirtz, Martin Henze,
Hossein Shafagh, Klaus Wehrle
Communication and Distributed Systems, RWTH Aachen University, Germany
{hummen, hiller, wirtz, henze, shafagh, wehrle}@comsys.rwth-aachen.de

ABSTRACT

6LoWPAN is an IPv6 adaptation layer that defines mechanisms to make IP connectivity viable for tightly resource-constrained devices that communicate over low power, lossy links such as IEEE 802.15.4. It is expected to be used in a variety of scenarios ranging from home automation to industrial control systems. To support the transmission of IPv6 packets exceeding the maximum frame size of the link layer, 6LoWPAN defines a packet fragmentation mechanism. However, the best effort semantics for fragment transmissions, the lack of authentication at the 6LoWPAN layer, and the scarce memory resources of the networked devices render the design of the fragmentation mechanism vulnerable.

In this paper, we provide a detailed security analysis of the 6LoWPAN fragmentation mechanism. We identify two attacks at the 6LoWPAN design-level that enable an attacker to (selectively) prevent correct packet reassembly on a target node at considerably low cost. Specifically, an attacker can mount our identified attacks by only sending a single protocol-compliant 6LoWPAN fragment. To counter these attacks, we propose two complementary, lightweight defense mechanisms, the *content chaining scheme* and the *split buffer approach*. Our evaluation shows the practicality of the identified attacks as well as the effectiveness of our proposed defense mechanisms at modest trade-offs.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection (e.g., firewalls)*

Keywords

Internet of Things; 6LoWPAN; Fragmentation; Hash chains

1. INTRODUCTION

6LoWPAN [23, 31] is an IETF-standardized IPv6 adaptation layer that enables IP connectivity over low power, lossy network links. It is envisioned as the main building block for a number of network scenarios in the Internet of Things including home automation, industrial control systems, and smart cities [27]. Accordingly, a wide range of applications in these scenarios employ 6LoWPAN for IP-based communication via standard or special-purpose upper layer protocols.

As its main task, 6LoWPAN adjusts IPv6 packets to the unique characteristics and requirements of wireless multi-hop communication between low-power devices. The variety of applications thereby requires 6LoWPAN to support both small-sized transmissions, e.g., for sensor data or control commands, and large transmissions, e.g., for firmware updates or security protocol handshakes [37, 24, 19].

To enable the transmission of large IPv6 packets over size-constrained link layer technologies such as IEEE 802.15.4 [26], 6LoWPAN provides fragmentation support at the adaptation layer. However, the design of the 6LoWPAN fragmentation mechanism renders buffering, forwarding and processing of fragmented packets challenging on resource-constrained devices. Specifically, malicious or misconfigured nodes may send duplicate or overlapping fragments. Due to the lack of authentication at the 6LoWPAN layer, recipients are unable to distinguish these undesired fragments from legitimate ones for packet reassembly. Moreover, reassembling nodes have to optimistically store fragments of a packet and rely on a timeout mechanism to discard incomplete packets. This, however, may cause the scarce memory of a node to be occupied with incomplete packets due to missing fragments. Thus, lossy links as well as malicious or misconfigured nodes can block the processing of newly received fragmented packets by spuriously occupying buffer resources.

Our contribution in this paper is the detailed security analysis of the 6LoWPAN fragmentation mechanism for networks that consist of resource-constrained devices. We identify two attacks that a malicious node can mount against the 6LoWPAN layer. First, an eavesdropping attacker can reactively prevent the successful processing of fragmented packets by duplicating an overheard fragment with the *fragment duplication attack*. Second, an attacker without overhearing capabilities can pro-actively block processing of any fragmented packet at the target node by sending a single 6LoWPAN fragment with the *buffer reservation attack*.

To protect resource-constrained devices against these attacks, we propose two complementing, lightweight mechanisms. The *content-chaining scheme* mitigates the fragment duplication attack by offering efficient *per-fragment* sender authentication. Moreover, the *split buffer approach* fosters competition for the scarce buffer resources between legitimate nodes and an attacker on a *per-packet* basis. Our

WiSec'13, April 17-19, 2013, Budapest, Hungary.

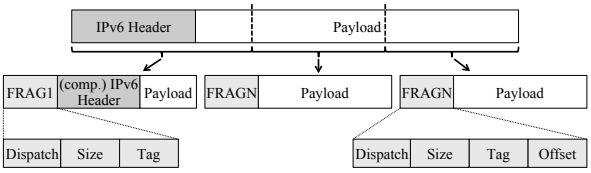


Figure 1: 6LoWPAN packet structure of a first fragment FRAG1 and subsequent fragments FRAGN.

packet discard strategy for the split buffer purges packets with suspicious sending behavior from the buffer in case of a buffer overload situation. Our evaluation shows that these mechanisms mitigate the identified attacks at low cost.

The structure of this paper is as follows. We first give a brief overview of the 6LoWPAN fragmentation mechanism in Section 2. We then describe the network scenario and the attacker model, and provide a detailed security analysis of the 6LoWPAN fragmentation mechanism in Section 3. Based on these findings, we introduce our complementary, lightweight countermeasures in Sections 4 and discuss their security considerations in Section 5. In Section 6, we show that our proposed approaches protect against the identified attacks and discuss their trade-offs. Finally, Section 7 discusses related work and Section 8 concludes our paper.

2. 6LoWPAN PACKET FRAGMENTATION

As a basis for our security analysis, we now give a brief overview of the 6LoWPAN fragmentation mechanism. We also discuss the packet routing mechanisms supported by 6LoWPAN and their implications on fragment forwarding.

2.1 Fragmentation Mechanism

The 6LoWPAN adaptation layer is located between the network and the link layer. It provides header compression and packet fragmentation functionality for IPv6 packets. In case of packet fragmentation, each 6LoWPAN fragment carries information that allows for in-place reassembly, even for out-of-order fragments. In contrast to regular IP fragments, 6LoWPAN fragments only include IP header information in the initial fragment of a packet.

When an IPv6 packet at a sending node exceeds the available link layer payload size, the 6LoWPAN fragmentation mechanism treats the (compressed) IPv6 packet as a single data field and iteratively segments this field into fragments according to the maximum frame size at the data link layer. Each fragment includes a fixed-size *fragment header*. The remaining space of the link-layer frame is iteratively filled with the IPv6 packet content (see Figure 1).

This process implies that only the first fragment (*FRAG1*) contains end-to-end routing information. Hence, a receiving node needs to correlate the remaining fragments (*FRAGN*) to the FRAG1 in order to derive IP-based routing or processing decisions for these fragments. To this end, 6LoWPAN fragments contain a *datagram tag* that is included in each fragment header and is unique per sender and fragmented packet. Thus, the datagram tag enables a receiving node to look up routing information for all fragments belonging to a fragmented packet after the FRAG1 has been received.

Each fragment also carries information that allows for in-place reassembly of fragmented packets at a receiving node. The *datagram size* of the unfragmented (and uncompressed) IPv6 packet enables a receiving node to reserve buffer space for reassembly of the whole packet. The *datagram offset* indicates the position of the current payload within the original IPv6 packet and thus the reassembly buffer.

2.2 Fragment Forwarding Mechanisms

6LoWPAN supports three routing mechanisms [31, 2]. *Mesh-under* routing offers packet forwarding based on link-layer routing schemes. In contrast, *route-over* delegates routing to the network layer on a per-packet basis, which *enhanced route-over* optimizes by applying forwarding decisions on a per-fragment basis after the FRAG1 is received.

With *mesh-under* routing, the 6LoWPAN layer prepends each fragment with a mesh routing header. This header contains the end-to-end source and destination link layer addresses. As the link layer routing scheme at a forwarding node can immediately use this information to derive a routing decision on a per-fragment basis, mesh-under routing is oblivious to packet fragmentation. As a result, individual fragments may take different paths towards the destination.

In contrast to mesh-under routing, *route-over* routing does not require additional header information and derives forwarding decisions at the network layer. To this end, a receiving node first reassembles the entire packet before passing the packet to the upper layers for processing. If the packet is destined for another node, the receiving node looks up the next hop in its IPv6 routing table and passes the packet to the 6LoWPAN layer for re-fragmentation. As forwarding nodes apply the routing decision on a per-packet basis, all fragments of a packet are sent along the same path.

To afford mesh-under-like forwarding efficiency, *enhanced route-over* [2] proposes an optimization of the route-over approach. Enhanced route-over derives forwarding decisions directly based on the IP header information in the FRAG1. It then stores the forwarding decision, forwards the FRAG1, and applies the same forwarding decision on reception of a FRAGN that belongs to the same IPv6 packet. Hence, while FRAGNs can be forwarded individually, they are transmitted along the same path, similar to route over.

We now proceed with the security analysis of the presented 6LoWPAN fragmentation and routing mechanisms.

3. SECURITY ANALYSIS

Our security analysis focuses on how an attacker can misuse the 6LoWPAN fragmentation and routing mechanisms in order to deny the correct processing of legitimate fragmented packets. Specifically, we focus on the challenging case of an *in-network, standard-compliant* attacker that directly exploits vulnerabilities of the 6LoWPAN protocol design and characteristics of the respective network scenario.

We make no assumptions regarding the attacker's hardware resources. Thus, our identified attacks are feasible even for resource-constrained devices that are similar or equal to the actual devices in the network. Furthermore, the apparently benign behavior at the 6LoWPAN layer makes the attacker and the attacks themselves hard to detect. As such, these attacks are complementary to research on network-external attacks such as jamming-based attacks [32, 38].

We now describe the network scenario and the attacker model as the basis of our security analysis. We then proceed with our discussion of the identified attacks and analyze the topological position of a node that can be targeted depending on the routing scheme and the location of the attacker.

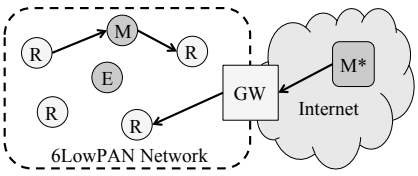


Figure 2: Network scenario with resource-constrained nodes (R) that connect to the Internet via a gateway (GW). The attackers Eve (E), Mallory (M), and Malice (M*) are marked in dark grey. Arrows indicate specific forwarding paths.

3.1 Network Scenario

In our security analysis, we abstract from specific device types, link layer technologies, and network topologies and only regard generic network characteristics. Still, our focus lies on resource-constrained devices. Hence, our network scenario consists of devices with only a few MHz of computational power and tens of kilobytes of RAM. These devices communicate over low-power wireless links and may optionally use security mechanisms provided at the link layer, e.g., based on network-wide keys. We assume the 6LoWPAN network to be connected by a gateway to a backbone infrastructure such as the Internet (see Figure 2).

Most notably, the ability of the resource-constrained devices in our network scenario to process and store fragmented packets for reassembly is very limited. For example, the default configuration of the Contiki operating system [14] for the Tmote Sky platform is restricted to the reassembly of only a *single fragmented packet at a time* with a maximum IPv6 packet size of 240 bytes.

3.2 Attacker Model

We distinguish between three different types of attackers: *Eve*, *Mallory*, and *Malice*. Eve and Mallory are both network-internal attackers who participate in the 6LoWPAN network. To join a network without link layer security, Eve and Mallory can simply be placed within radio range of the target network. In case of a protected network, an attacker must first gain admission to the network, e.g., by extracting the security keys from a legitimate node [20, 1].

Both, Eve and Mallory, participate in the routing structure and thus can send messages to any node in the network. However, with respect to the forwarding path of specific fragmented packets, they are situated in different network locations (see Figure 2). Eve is located *besides* the forwarding path of the fragmented packets. Thus, she can overhear the communication channel and send packets in reaction to overheard messages. Mallory is located *on* the forwarding path. Hence, she has Eve's capabilities and can also delay, reorder, alter or simply drop legitimate packets. The capabilities of Mallory allow her to mount at least the attacks that are viable for Eve. Thus, we do not mention her explicitly in our security considerations when discussing Eve.

In contrast to Eve and Mallory, Malice is located outside the 6LoWPAN network and has significantly more resources than the resource-constrained nodes. This enables her to simply flood a resource-constrained node with numerous large packets [21]. The fragmentation of these packets at the gateway further amplifies this attack by increasing the number of packets that the target node has to process.

To protect against such flooding-based attacks from Malice, the gateway may employ authenticated tunnels to external hosts as well as rate limitation for large packets from authenticated sources. Authenticated tunnels enable the gateway to exclude external hosts from communication if they do not behave correctly. Furthermore, rate limitation at the gateway prevents the 6LoWPAN network or a single 6LoWPAN node from being overloaded due to the vast difference in network resources. However, these mechanisms at the gateway still leave the 6LoWPAN network vulnerable to network-internal attackers. We therefore focus our discussion on attacks that Eve and Mallory can mount.

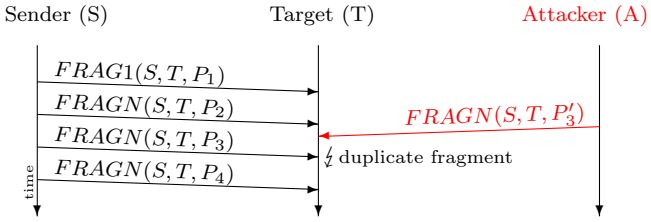


Figure 3: Packet diagram depicting the fragment duplication attack. A target must decide which fragment payload (P_3 or P_3') to use during reassembly.

3.3 Fragment Duplication Attack

The fragment duplication attack leverages the fact that a recipient cannot verify at the 6LoWPAN layer if a fragment originates from the same source as previously received fragments of the same IPv6 packet. Thus, the recipient cannot distinguish legitimate fragments from spoofed duplicates at the time of reception. Instead, it has to process all fragments that appear to belong to the same IPv6 packet according to the sender's MAC address and the 6LoWPAN datagram tag.

Eve can exploit this fact to *selectively* block the reassembly of specific fragmented packets at a target node. For example, she may aim at preventing secure communication by blocking handshake packets of the DTLS protocol. To do so, she inspects the wireless medium for fragments that contain a DTLS message type. She then injects spoofed FRAGNs with random payload and a fragment header that links her fragments to the legitimate 6LoWPAN packet, as illustrated in Figure 3. As the target node cannot distinguish spoofed and legitimate FRAGNs, it cannot decide which fragments to use during packet reassembly at the 6LoWPAN layer.

In general, Eve can block the delivery of *any* fragmented IPv6 packet in her vicinity by injecting FRAGNs for each observed packet. In addition, higher layer protocols may retransmit lost application data in order to ensure reliable delivery, e.g., for confirmable messages in CoAP [36]. Eve's blocking of retransmitted packets then further depletes the energy resources of the forwarding and the target nodes.

Due to the complexity of dealing with duplicate fragments, the 6LoWPAN standard suggests to drop corrupt IPv6 packets. This, however, allows Eve to force her target to drop fragmented packets by sending a single duplicate FRAGN.

Upper layer information of the *reassembled packet* (e.g., message authentication codes) could be used to identify the correct fragment combination for packets with duplicate fragments. However, such an approach shows significant shortcomings. Most importantly, it requires the recipient to store all received fragments and to reassemble them after each fragment has been received at least once. Spoofed duplicates thus cannot be detected early during fragment reception and may overload the scarce buffer space at the receiver.

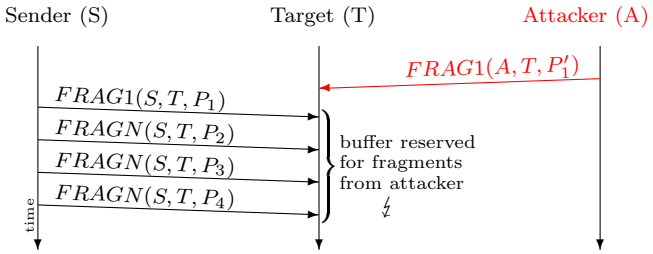


Figure 4: Packet diagram illustrating the buffer reservation attack. After the attack, the reassembly buffer of the target node is occupied by attacker fragments until the reassembly timeout expires.

3.4 Buffer Reservation Attack

The buffer reservation attack targets the scarce memory of resource-constrained nodes and leverages the fact that the recipient of a fragmented packet cannot determine a-priori if all fragments will be received correctly. Hence, a receiving node must optimistically reserve buffer space for the reassembly of the complete packet as indicated in the 6LoWPAN header. Other fragmented packets are dropped by the recipient if the reassembly buffer is already occupied. As the buffer reservation attack affects an individual reassembly buffer, the effort for an attacker to mount this attack grows linearly with the number of buffers at the target node. For our discussion of the buffer reservation attack, we assume that the target node operates with a single reassembly buffer as is the case for the Contiki operating system.

6LoWPAN defines a reassembly timeout of up to 60 seconds in order to handle fragment loss on the communication path. This timeout aims to prevent the reassembly buffer from being occupied by an incomplete packet indefinitely. Hence, when this timeout expires on a reassembling node, it must drop an incomplete packet from its reassembly buffer in order to free memory for new fragmented packets. Eve can exploit this mechanism to mount a DoS attack against memory-constrained nodes by maliciously reserving the reassembly buffer with incomplete packets.

To mount a buffer reservation attack, Eve generates a *single FRAG1* with arbitrary payload and sends it towards her target as shown in Figure 4. If the buffer of the target is not yet occupied by another fragmented packet, the received FRAG1 reserves the buffer for the reassembly of Eve's fragmented packet. Eve now either does not send the remaining FRAGNs or releases them sporadically in order to occupy the buffer resources until the reassembly timeout expires. During this time, no additional fragmented packets can be processed by the target node. Furthermore, the target node cannot distinguish Eve's attack fragments from fragments of benign senders with intermittent delays induced by the low-power characteristics of 6LoWPAN networks.

To *continuously* mount the buffer reservation attack, Eve either needs to constantly send FRAG1 fragments or she has to time her attack according to the reassembly timeout value of the target node. In case of a timing attack, the next FRAG1 must be received by the target immediately after the timeout of the previous attack has expired. To learn the exact reassembly timeout value, Eve first has to probe the target's timeout in preparation of her attack.

When probing the target, Eve sends an attack FRAG1 followed by a sequence of complete fragmented packets that trigger a response from the target. For example, she may send ICMP echo requests with a large data field that causes packet fragmentation. If the reservation attack succeeds, Eve only receives replies from the target node for fragmented packets that arrived after the reassembly timeout expired. Hence, Eve can estimate the reassembly timeout of the target node by measuring the time between the FRAG1 and the first reply. Eve is then able to mount a continuous reservation attack by sending single FRAG1s and possibly sporadic FRAGNs according to the probed timeout value.

3.5 Susceptibility of the Routing Schemes

The prerequisite for the identified fragment duplication and the buffer reservation attacks is that the target node reassembles fragmented packets for forwarding purposes or as the 6LoWPAN destination. Hence, the routing mechanism used in the 6LoWPAN network determines which resource-constrained nodes Eve can target with her attacks.

For *route-over*, Eve can interfere with the fragment processing of her one-hop neighborhood. This is because each node reassembles fragmented packets to derive routing decisions at the network layer. Thus, both attacks enable Eve to block all fragmentation-based communication that traverses the target nodes. However, Eve cannot target the reassembly of nodes that are located topologically farther away as her immediate neighbors do not forward attack packets.

In case of *enhanced route-over* or *mesh-under* routing, forwarding nodes do not reassemble fragmented packets, but directly forward attack fragments towards the destination. This enables Eve to mount the identified attacks against arbitrary 6LoWPAN destinations without topological restrictions. Hence, she can target *every* node in the 6LoWPAN network instead of only her one-hop neighborhood.

4. SECURE 6LoWPAN FRAGMENTATION

In this section, we propose lightweight security mechanisms that protect resource-constrained nodes against the fragmentation-based attacks we identified in Section 3. We note that nodes could protect themselves against the fragment duplication attack if the 6LoWPAN layer allowed them to distinguish legitimate and spoofed attack fragments on a *per-fragment* basis. We achieve this property with our *content-chaining scheme* that cryptographically binds the content of a fragmented packet to its FRAG1.

We also introduce a *split buffer approach* with fragment-sized buffer slots in order to enable processing of legitimate fragmented packets in spite of malicious nodes that pretend to require reassembly buffer resources during the buffer reservation attack. We combine this split buffer approach with a packet discard strategy that disposes of fragmented packets with suspicious sending behavior in case of a buffer overload situation. We now present a detailed description of our proposed mechanisms and refer the reader to Section 6 for the discussion of the specific trade-offs.

4.1 Content Chaining Scheme

Resource-constrained nodes could defend against the fragment duplication attack if they were able to *identify the sender* on a per-fragment basis. However, even networks with link layer security based on network-wide keys only offer group authentication. Hence, a network-internal attacker can still spoof fragments. In contrast, pairwise keys at the link layer would prevent spoofing of link layer addresses. As route-over-based routing mechanisms use the

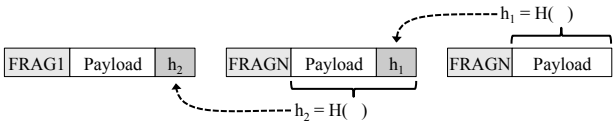


Figure 5: Example of a content chain for a packet consisting of three fragments.

link layer source address in combination with the datagram tag to identify the original packet of a fragment, this would help mitigating the fragment duplication attack. However, the necessary key management is non-trivial and would require central coordination [4], public key cryptography, e.g., a Diffie-Hellman key exchange, or alternative approaches such as probabilistic key pre-distribution [16, 6]. Furthermore, pairwise keys would be required on an end-to-end basis for mesh-under routing as here the mesh header source address is used to identify the original packet of a fragment.

To avoid the overhead of a pairwise-key management, we propose a *content-chaining scheme* that binds the content of a fragmented packet to its FRAG1 instead of binding fragments to cryptographic sender identities. To this end, the legitimate sender adds an authentication token to each fragment during the 6LoWPAN fragmentation procedure. This allows the recipient to cryptographically verify the link between fragments at the time of reception and to discard maliciously duplicated fragments early on the forwarding path.

4.1.1 Content Chain Construction

While the 6LoWPAN standard does not define the sending order of a fragmented packet, *in-order transmission* starting from the FRAG1 has recently been recognized as highly advisable, especially in networks employing (enhanced) route-over [2]. The design of our *content chaining scheme* takes advantage of this new development. Specifically, the legitimate sender cryptographically commits to the content of a fragmented packet in the corresponding FRAG1 by means of a *content chain*. Content chains are based on the concept of hash chains [17, 5], a lightweight, efficient mechanism to authenticate the sender of a data stream of finite length. Thus, they naturally fit the properties of fragmented packets when treating each packet as a fragment stream.

The elements h_i of a hash chain are generated by iteratively applying a cryptographic hash function $H(\cdot)$ to the output of the previous iteration: $h_i = H(h_{i-1})$. For the first iteration, a (random) seed value h_0 is used as input. The last token $h_n = H^n(h_0) = H(H(\ldots H(h_0)))$ is referred to as the anchor element of the hash chain. Hash chains are used in reverse order of their generation, i.e. starting with h_n, as the one-way property of the hash function then prevents others from computing undisclosed tokens that are closer to the seed value. Thus, a token represents a cryptographic commitment to all previous tokens of the hash chain.

However, simply appending the hash chain elements h_i to 6LoWPAN fragments as one-time tokens does not suffice to bind these fragments to the legitimate sender as FRAGNs may be received out-of-order on the forwarding path. Specifically, if an attacker received an out-of-order fragment containing token h_{j-k} $(k > 0)$, she could compute the token h_j of a previous fragment that has not yet been received on the forwarding path. The attacker could then use this token to create a valid duplicate fragment. Hence, we extend the general structure of a hash chain and include the actual fragment content in the hash chain generation in order to enable the secure verification of out-of-order fragments.

When generating a content chain, the legitimate sender uses the payload of the last $FRAGN$ as the seed value for the hash chain. He then appends the resulting content token to the previous $FRAGN$ and computes the hash digest over the fragment content including the appended token (see Figure 5). After iteratively performing this procedure over all packet fragments, the FRAG1 contains a token that commits to the overall packet content. Once the content chain construction has finished, the legitimate sender transmits each fragment with its respective content token.

4.1.2 Content Chain Verification

When a node receives a FRAG1 with a content token, it processes the packet normally and stores the contained token for verification purposes of subsequent FRAGNs. For each received FRAGN, it then validates the current fragment by computing the hash over the received packet content. If the computed hash digest matches the stored token, the verification has been successful and the stored token is replaced by the currently verified one. Otherwise, the fragment is regarded as spoofed and can be dropped immediately.

As the FRAG1 transitively commits to the subsequent fragments, target nodes are able to detect spoofed fragments after receiving the FRAG1. This prevents an attacker from interfering with the packet reassembly by sending duplicates.

4.1.3 Processing Out-of-order Fragments

Content chaining enables a verifying node to cryptographically determine a single valid 6LoWPAN fragment combination for an IPv6 packet despite the reception of malicious duplicate fragments. Notably, our content chaining scheme is not required to be robust against token loss because a lost fragment invalidates the entire packet according to the 6LoWPAN standard. However, a verifying node may not be able to verify out-of-order FRAGNs directly on reception because the previous fragment, and thus the commitment to the content of the current fragment, may still be missing.

Hence, when processing out-of-order FRAGNs, a verifying node follows a simple policy. First, it only forwards fragments that have been verified successfully. Second, it stores out-of-order FRAGNs without prior verification until all previous FRAGNs have been received and verified.

This simple policy enables a malicious node to fill the reassembly buffer of a verifying node with fragments that appear to be out-of-order fragments of a legitimate packet. To counter such an attack, the verifying node discards the fragment with the largest datagram offset when reaching a buffer overload situation. This fragment has the largest distance from the last correctly verified fragment. Since a node only forwards already verified fragments, this fragment is least likely to be the result of one-hop reordering.

4.1.4 Implementation Considerations

The hash function used in our content chaining scheme requires memory resources for its implementation as well as computational resources per token generation and verification. To decrease these overheads, we propose to construct the hash function based on a block cipher-based one-way compression function [3] with a length-padded Merkle-Damgård construction [30, 10]. This allows us to use the hardware acceleration support for the AES block cipher that

many embedded platforms provide due to built-in security functionality of the IEEE 802.15.4 radio interface.

Content tokens generate storage and transmission overheads. We propose to decrease these overheads by truncating tokens to 8 bytes. Due to the short validity period of a token, i.e., the reassembly timeout, the remaining cryptographic strength prevents an attacker from calculating a valid pre-image for a given token. Moreover, each hash operation is salted with the actual fragment content. This and the relative shortness of content chains (an IPv6 packet of 1280 bytes requires less than 20 tokens) makes our construction more robust against cycles, i.e., re-occurring elements, than the simple hash chain structure described above [5].

4.2 Split Buffer Approach

The buffer reservation attack allows to block the reassembly buffer of a target node for the timespan of the reassembly timeout at exceptionally low cost, i.e., with a single well-timed fragment. In this section, we propose mechanisms to increase these costs for an attacker such that she has to continuously send complete fragmented packets in short bursts in order to prevent legitimate packets from being processed at the target node. In this case, the buffer reservation attack resembles a flooding attack. Hence, the attacker does not benefit significantly from sending fragmented packets over unfragmented packets and must have sufficient resources to mount a flooding-based DoS attack against the target node.

4.2.1 Fragment-sized Buffer Slots

We observe that a single reassembly buffer forces a node to make a decision whether to replace fragments of a partially received packet with a new fragmented packet as soon as the first new fragment arrives. This prevents the node from optimistically storing fragments from multiple senders and deferring the decision which packet to discard to a point when an *actual* buffer overload situation is reached. Instead, it suffices for an attacker to *pretend* that she will use the available buffer resources by sending a single fragment.

Even with multiple reassembly buffers, the reassembling node would allocate memory resources as indicated in the 6LoWPAN header for each buffer. Thus, an attacker could occupy all buffer resources by sending multiple incomplete packets with an indicated high packet size. However, if a node stored *individual fragments of multiple packets* in its reassembly buffer, legitimate and malicious packets would compete for the available buffer resources based on the actually used buffer space. An attacker would then have to follow up on her pretense by transmitting further fragments.

To enable direct competition for the buffer resources between legitimate nodes and an attacker, we propose to split the reassembly buffer into *fragment-sized buffer slots*. Each slot has the maximum size of a 6LoWPAN fragment for a given link layer. Buffer slots are filled until either a packet has been fully received or an overload situation is reached. In case of a complete packet, the reassembling node assembles the packet in-order in the buffer and processes the packet normally. In case of a buffer overload situation, the node has to decide which packet to discard. To this end, the reassembling node can base its decision on the *observed sending behavior* for packets located in the split buffer.

4.2.2 Packet Discard Strategy

We propose a discard strategy for packets in the split buffer that is based on *per-packet scores*, capturing the extent to which a packet is completed along with the continuity in the sending behavior. In case of a buffer overload situation, the node then discards the packet with the lowest score. If two or more packets share the lowest score, the selection is performed randomly between these packets.

We identify three fundamentally different sending behaviors that an attacker may show during the buffer reservation attack. First, the attacker may send only the first fragment and skip the remaining ones. This sending behavior requires the least commitment of energy resources from an attacker. However, a long gap after the first fragment would indicate a loss of the remaining fragments in case of normal sending behavior. Thus, a reassembling node should preferably process other fragmented packets in an overload situation even before the reassembly timeout expires.

An attacker may also choose between two more sophisticated sending patterns. On the one hand, an attacker may immediately send all but a few fragments in a short burst in order to occupy as many buffer resources at the target node as possible. She may then transmit the remaining fragments shortly before the reassembly timeout expires. However, the long gap after the first fragment burst again indicates a loss of the remaining fragments. On the other hand, an attacker may stretch fragment transmission across the timespan of the reassembly timeout. With this pattern, an attacker appears most legitimate compared to the other behaviors, but only fills the buffer resources of the target node slowly.

Our *discard strategy* for the split buffer takes advantage of these observations. It forces an attacker to send complete packets in short bursts during the buffer reservation attack.

Percentage of Completion. We first show how to prioritize packets, that are sent in a short burst, during a buffer overload situation. To this end, a node scores each packet in the split buffer based on the *percentage of completion*. Short packet bursts increase this score quickly. Likewise, small packets that only require few buffer resources promptly get a high score as each received fragment contributes significantly to the packet completion. In contrast, attack fragments that are received at a low rate increase the score slowly and only add little to the score as an attacker must send large packets to cover (a substantial portion of) the reassembly timeout. Hence, in an overload situation, short packets as well as large packets that are sent in bursts are likely to have a higher score than slowly arriving attack packets. This renders low sending rates unattractive for an attacker.

Sending Behavior. An attacker may also change her sending behavior after she occupies buffer resources at the target with a high score. She may, e.g., stop transmitting after an incomplete burst of fragments. To penalize such a change in sending behavior, we additionally incorporate the time domain in the discard strategy. Specifically, we consider the *average elapsed time between two consecutive fragments* (a) of a packet and the *elapsed time since the last fragment* (l). Reassembling nodes penalize senders by reducing the score of a packet if the currently elapsed time l differs significantly from the expected time a. Nodes therefore store and update the time values l and a after each fragment reception.

To reflect a change in sending behavior in the packet score computation, we introduce a window w around the expected fragment reception time a. As long as the sending behavior does not change considerably ($a - w < l < a + w$), the score is calculated as before. However, if the reassembling node

receives a fragment earlier than expected ($l <= a - w$), it decreases the packet score by half. Likewise, if the fragment arrives later than expected ($l >= a + w$), the score is halved for each presumably missing fragment ($\lfloor l/a \rfloor$):

$$score_{i+1} = \begin{cases} \frac{\text{fragment bytes}}{\text{total bytes}} & \text{if } i = 0 \\ score_i + \frac{\text{fragment bytes}}{\text{total bytes}} & \text{if } a - w < l < a + w \\ \frac{score_i}{2^{\max\{1;\lfloor l/a \rfloor\}}} & \text{else} \end{cases}$$

The score of a packet in the split buffer increases proportionally to the contribution of each fragment to the overall packet completion. If the sending behavior for a packet changes significantly, the score decreases depending on the intensity of this change. The window parameter w thereby allows to calibrate the maximum tolerated change in sending behavior to the specific network characteristics.

When a buffer overload situation occurs, the reassembling node has to compare the score of the packets in the split buffer *at the time of the discard decision*. To consider the elapsed time since the last fragment reception, the node computes the current score similarly as described above:

$$score_{compare} = \begin{cases} score & \text{if } a - w < l < a + w \\ \frac{score}{2^{\max\{1;\lfloor l/a \rfloor\}}} & \text{else} \end{cases}$$

The node then discards the packet with the lowest score or randomly chooses between packets with the lowest score.

With our proposed split buffer approach, a reassembling node prioritizes competing fragmented packets that are sent in short bursts. Furthermore, significant changes in sending behavior are severely penalized. This forces an attacker to effectively flood the target node with short fragment bursts and considerably limits the practicality of the attack.

At the same time, our split buffer approach allows for a more efficient use of the reassembly buffer resources for legitimate communication than a single or multiple reassembly buffers of the same size. As buffer resources are assigned on a per-fragment basis, this even allows a node to process interleaved packets that, combined, would otherwise exceed the overall buffer resources. We note that legitimate nodes that slowly send large fragmented packets may be at a disadvantage in network scenarios that rely heavily on the transmission of large fragmented packets and where interleaved packet reception occurs often. This is a trade-off of our split buffer approach for the gained robustness against the buffer reservation attack. Still, our split buffer approach effects a considerable, network-wide throughput increase in such network scenarios as a slowly sending node would block nodes with high sending rates for a notable amount of time.

5. SECURITY CONSIDERATIONS

We now identify and briefly discuss attacks that adversaries Eve and Mallory can mount against our proposed approaches. As noted earlier, the capabilities of Mallory allow her to mount at least the attacks that Eve can mount.

Impact of an on-path attacker. Mallory may buffer legitimate fragments before forwarding them towards the 6LoWPAN destination. This would allow her to replace the legitimate fragment payload and to compute a valid content chain for the altered packet content. However, the token included in the FRAG1 only commits to a single fragment combination. Hence, her attack does not result in duplicate fragments. Instead, it resembles dropping of the entire original packet and creating a new one. These, however, are inherent capabilities of an on-path attacker. Furthermore, we consider the integrity protection of the overall packet a task for upper layer protocols.

Likewise, Mallory may forward legitimate fragmented packets with varying artificial delays between the individual fragments during a buffer reservation attack. As a result, the forwarded packets would have a lower score than her attack packets. However, she would achieve the same result if she dropped the legitimate packets instead of forwarding them.

Content chaining and FRAG1 spoofing. Eve may overhear a legitimate FRAG1 with a valid token and generate spoofed FRAG1s that include this token. Eve may then inject these spoofed FRAG1s on the forwarding path. To enforce a decision for a single FRAG1 in case of duplicates, a node only considers the first received FRAG1. Other FRAG1s with a matching source address and datagram tag are dropped immediately. As the direct transmission between two (legitimate) nodes is typically faster than through the off-path attacker Eve, this prevents her from attacking the recipient of the overheard FRAG1 with spoofed FRAG1s. However, Eve may be able to attack the 6LoWPAN destination in a mesh-under-based network if she knows a faster forwarding path to the destination node. Notably, in case of (enhanced) route-over, Eve must be in direct communication range of a next hop on the forwarding path and transmit her FRAG1s prior to the legitimate forwarder. Especially for enhanced route-over routing, this considerably limits Eve's capability to mount an attack as nodes immediately forward the legitimate FRAG1s after reception.

Content chaining and a reordering attacker. Eve may take advantage of the fact that out-of-order fragments cannot be validated immediately at a verifying node. To this end, she may send fragments with a large offset in order to occupy buffer resources until all previous fragments have been received and the fragment is discarded due to an invalid content chaining token. However, the discard policy of the content chaining scheme would cause such fragments to be dropped first in case of a buffer overload situation. Hence, Eve may send fragments that are close to the current fragment offset of the legitimate packet. Still, these fragments are discarded upon reception of the legitimate fragments.

Split buffer and unfair competition. Eve may try to maintain a high packet score at the target node by sending large fragments at a low rate. As soon as she detects that another node sends a fragmented packet, she may change her sending behavior to a high sending rate for the remaining fragments in order to block the majority of the target's buffer slots. We account for such behavior in our packet discard strategy by penalizing senders that suddenly change the sending rate. As the packet score is halved for each fragment that is received early, Eve's score would decrease significantly. This makes newly received legitimate packets competitive despite the lack of an initial score.

6. EVALUATION

For our evaluation, we implemented our proposed defense mechanisms for the Contiki operating system version 2.5. We used Tmote Sky motes that are equipped with an 8 MHz MSP430 microcontroller, 10 kB of RAM, 48 kB of ROM, and an IEEE 802.15.4 radio interface as our evaluation platform. As the network setup depends on the evaluated property, we describe the network topology in each section individually.

Regarding Contiki, we used the standard configuration where possible. We only decreased the number of neighbors in the RPL neighborhood table from 20 to 6 in order to have sufficient memory resources to evaluate the behavior and overheads of our proposed mechanisms for IPv6 packets of up to 1280 bytes[1]. While this change allowed us to increase the reassembly buffer from 240 to 1280 bytes, it limits the maximum connectivity of a node. As we expect deployment scenarios to require tailored trade-offs, we highlight the results for the default size in the discussion of our results.

As the hash function for the content chaining scheme, we implemented a Davies-Meyer one-way compression function [3] with a length-padded Merkle-Damgård construction. Hence, we could leverage the AES hardware support of the CC2420 radio interface when computing hash digests.

For our evaluation, we did not consider explicit overheads due to link layer security operations such as encryption and decryption. However, we considered implicit overheads resulting from the *maximum length* of the security header at the link layer by decreasing the available 6LoWPAN payload size by 21 bytes. As a result, FRAG1s contained up to 88 bytes of IPv6 header information and payload, whereas FRAGNs contained between 1 and 72 bytes of payload. Hence, our results indicate worst case overheads for our proposed mechanisms due to an increased number of fragments.

6.1 Defense Against the Identified Attacks

We now show the practical existence of our identified attacks and the effectiveness of our defense mechanisms.

Fragment Duplication Attack. As a proof of concept for the fragment duplication attack, we implemented a simple sender that transmits a constant stream of fragmented UDP packets. To simulate the behavior of a spoofing attacker Eve, this node *additionally* sends one fragment of each legitimate packet with an altered 6LoWPAN payload. A second node receives packets and counts the correctly received packets.

During our evaluation, we ran two different configurations on both nodes: one with an unmodified Contiki implementation and the other one additionally using our content chaining scheme. The sender periodically transmitted 100 fragmented packets of 240 bytes, each consisting of 4 legitimate fragments and 1 duplicate attack fragment.

With an unmodified Contiki, the receiver experienced complete packet loss with a packet delivery rate (PDR) of 0 % at the UDP layer. For each received packet, the receiver detected an invalid UDP checksum as legitimate packet content was overwritten during the attack. In contrast, our content chaining scheme achieves a PDR of 100 %. Each received fragment was verified correctly. The number and the order of spoofed fragments do not influence these results as our content chaining scheme discards unverified fragments first if a buffer overload situation arises. Hence, we conclude that the content chaining scheme effectively mitigates the otherwise viable fragment duplication attack.

Buffer Reservation Attack. To confirm the existence of the buffer reservation attack and the effectiveness of our split buffer approach, we consider a network setup consisting of three nodes: a sender, an attacker Eve, and a target node that also denotes the destination of the legitimate packets.

The buffer reservation attack only succeeds if the attacker reserves buffer resources at the target node *before or while* the legitimate packet is received. Hence, we evaluated the attack for different relative reception times of legitimate and attack packets at the target node. We analyzed the PDR at the UDP layer for a sender who transmits packets 500 ms before, simultaneous to, or 500 ms after the first attack fragment is sent. While the -500 ms offset allows for legitimate transmissions without malicious buffer reservation, the 500 ms offset represents a situation where the attacker successfully occupies the desired resources of the reassembly buffer. For simultaneous transmissions, the order of reception for legitimate and attack packets is not pre-determined.

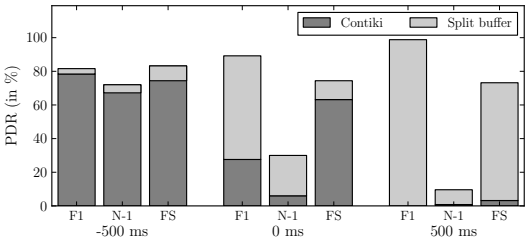


Figure 6: PDR for legitimate packets at the target node during the buffer reservation attack depending on the sending behavior and the sending offset.

We also analyzed the following three sending behaviors of an attacker: i) exclusive transmissions of FRAG1s (F1), ii) fragment bursts excluding the last FRAGN (N-1), and iii) fragment spreading across the reassembly timeout (FS). Moreover, we considered packet sizes of 240 and 1280 bytes for the sender. However, as the differences in the results are negligible in case of no protection and further improve for 240 byte packets when using our split buffer approach, we only discuss the results for packets of 1280 bytes.

To compare an unmodified Contiki and the split buffer approach, the target node was configured with these two functionalities respectively. For the split buffer approach, the window value w was set to 250 ms. We measured the PDR at the target node for 10 runs with 25 legitimate packets for each combination of the above configurations.

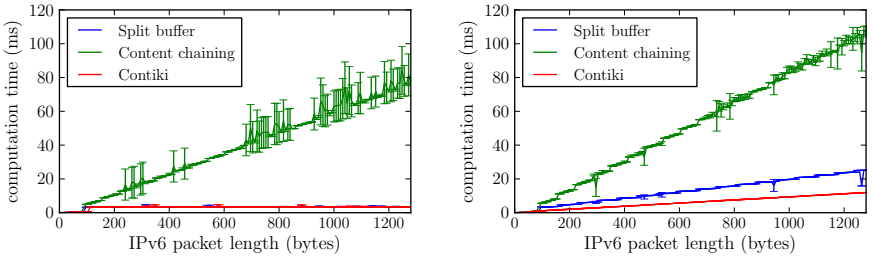
With an unmodified Contiki, the attack succeeds for all attack behaviors if the first attack fragment is received before the legitimate packet. In these cases, the PDR dropped as low as 0 % (see F1 for 500 ms case in Figure 6). In contrast, with our split buffer, the PDR increased up to 98 % depending on the attack behavior. Notably, the attacker has to send packets with a large number of fragments in short bursts in order to decrease the PDR with our split buffer approach (see N-1 cases in Figure 6). However, in these cases, our packet discard strategy significantly decreases the score of the attack packet once the delayed last fragment is detected as a deviation from the previously observed sending behavior. As a result, attack fragments are purged quickly from the split buffer when a new packet is received.

From these results, we conclude that the buffer reservation attack is viable against an unprotected 6LoWPAN layer even for a tightly resource-constrained attacker. In contrast, our split buffer approach forces an attacker to send short successions of large numbers of fragments during the attack.

6.2 Run-time Performance

To evaluate the run-time performance of our proposed approaches, we first analyzed the cryptographic *per-fragment*

[1]The IPv6 standard [11] requires every link to have a maximum transmission unit of 1280 bytes or greater.



(a) sending node (b) receiving node

Figure 7: Processing time at the 6LoWPAN layer. The error bars denote the standard deviation.

Mechanism	ROM	RAM
Contiki	37788	8478
Content chaining	41108 (+8.79%)	9402 (+10.90%)
Split buffer	40052 (+5.99%)	9014 (+6.32%)
All combined	41750 (+10.48%)	9502 (+12.08%)

Table 1: ROM and RAM requirements for our proposed approaches in byte. Numbers in brackets denote added overhead to Contiki.

overhead of the content chaining scheme on a single node. We then examined the computational *per-packet* overheads of the content chaining scheme and the split buffer approach.

For the analysis of the per-fragment overhead, we ran 1000 measurements for fragment payload sizes ranging from 8 to 72 bytes respectively. The average computation time for a single content token, i.e., generation or verification, increases from 0.87 ms to 5.22 ms for growing payload sizes (see Appendix Figure 8). Notably, the processing time for 16 byte inputs increases step-wise compared to 8 byte inputs, although the hash function operates on 16 byte blocks. This is because the hash input has to be extended by an additional block if the length padding information does not fit into the last input block. Moreover, we observed outlier hash operations with run-times above 30 ms and faulty hash digests in about 2 % of our measurements. The number of erroneous operations increased with a growing payload size and thus a rising number of iterative AES operations. During these erroneous operations the hardware interface constantly reported as busy via the *ENC_BUSY* flag.

To evaluate the content chaining and the split buffer overheads, we measured the packet processing time at the 6LoWPAN layer on a sending and on a receiving node. As we were interested in the expected results without hardware errors, we only considered measurements without erroneous tokens. Specifically, we analyzed the first 10 error-free measurements for each IPv6 packet size ranging up to 1280 bytes.

As shown by the largely overlapping results for the split buffer approach and for the unmodified Contiki in Figure 7(a), the split buffer approach does not impact the performance of a sending node. However, it adds a small overhead of up to 13.19 ms to the packet processing on a receiving node compared to an unmodified Contiki (see Figure 7(b)). The overhead of the content chaining scheme mainly stems from the construction of the content chain at the sending node and the aggregated token verification for all fragments of a packet at the receiving node. As a result, content chaining adds a maximum of 64.22 ms at the sender and of 95.23 ms at the receiver to the packet processing for packets of 1280 bytes. The performance overhead for the content chaining scheme on the receiver side is higher than on the sender side because content chaining also uses the buffer management functionality of the split buffer approach for handling out-of-order packets. Thus, the overall content chaining overhead also includes the majority of the performance overhead of the split buffer approach. Furthermore, the receiving node additionally has to search for unverified out-of-order fragments in the split buffer in case of fragment reordering.

The computational overhead of a forwarder is similar to the overhead on a receiver in case of enhanced route-over routing, and largely resembles the sum of a sender and a receiver for a route-over-based forwarder. A mesh-under-based forwarder is oblivious to packet fragmentation.

To avoid the overhead of the content chaining scheme in network scenarios without a misbehaving node, our scheme can also be implemented with a default-off policy and be switched on-demand as described in Appendix A.

6.3 Packet Overhead

The *split buffer approach* is a purely local mechanism and, thus, does not require the transmission of additional information. In contrast, the *content chaining scheme* adds 8 bytes per fragment to the overall packet transmission.

To evaluate the packet overhead, we inspected a FRAG1 and a FRAGN with maximum length for standard 6LoWPAN fragments as well as for fragments containing content tokens with the *wireshark* tool. We then extrapolated the gathered information for higher IPv6 packet sizes.

Notably, unfragmented packets do not contain content token information. Likewise, packets of only 2 fragments result in a token overhead that is as low as 8 bytes because the last FRAGN does not carry token information. Protecting the 6LoWPAN fragmentation of an IPv6 packet of 1280 bytes requires additional 254 bytes. This overhead results from the content tokens and from additional fragments caused by the decreased per-fragment payload space. To put these numbers into perspective, this overhead denotes 11.65 % of the overall packet transmission. This is because an IPv6 packet of 1280 bytes already requires a total transmission of 2180 bytes due to link layer and 6LoWPAN overheads, even without the content chaining scheme.

Overall, the moderate packet overhead increases linearly with the number of fragments in attack scenarios and can otherwise be avoided as described in Appendix A.

6.4 RAM and ROM Overhead

To derive RAM and ROM estimates for our proposed approaches, we analyzed the binary of a simple 6LoWPAN-enabled UDP application for four different configurations with the *msp430-size* tool. While the first binary contains an unmodified 6LoWPAN stack, the other three binaries additionally include i) the content chaining scheme, ii) the split buffer approach, and iii) a combination of both mechanisms.

As shown in Table 1, the ROM overhead of the split buffer amounts to 5.99 % of the Contiki base overhead, whereas content chaining generates 8.79 % overhead. About 530 bytes of the latter result from the implementation of our crypto-

graphic primitive. Content chaining and the split buffer approach use the same buffer management functionality. This is reflected by the low additional overhead for the combined mechanisms compared to the individual overheads. Notably, while not optimized for minimum ROM overhead, our implementation results in less than 4 kB of code.

With respect to RAM, our content chaining scheme and the split buffer approach require 10.90 % and 6.32 % additional memory, respectively, when compared to the Contiki base overhead. For the split buffer approach, this overhead stems from the need to over-provision each buffer slot such that it can hold a 6LoWPAN fragment of maximum length including 6LoWPAN header information, i.e., 81 bytes. As a result, the memory overhead for a reassembly buffer that supports IPv6 packets of 1280 bytes increases by 18.13 %. The remaining overhead results from additional per-packet management information required for our packet discard strategy and for maintenance of the individual buffer slots. This overhead, however, is not only a trade-off for a gain of security. It additionally enables a node to process interleaved legitimate packets during normal operation. Content chaining primarily adds an overhead of 8 bytes for the last verified token to this per-packet management information.

7. RELATED WORK

For our discussion of related work, we distinguish the following two research directions: i) previously identified fragmentation-based attacks and proposed protection mechanisms and ii) existing hash chain schemes.

Packet fragmentation has previously been identify as a potential security risk for today's IP-based communication as well as for 6LoWPAN-enabled networks. Regarding today's IP communication, a large number of fragmentation-based attacks have been identified [39, 7, 8]. However, these attacks commonly focus on deficiencies of the respective IP protocol implementation, e.g., for IDS or firewall evasion, or for DoS purposes [34]. Our work, on the contrary, focuses on inherent design-related issues of the 6LoWPAN layer that emerge when using the 6LoWPAN fragmentation mechanism in resource-constrained network environments.

Recently, Gilad et al. [18] discovered an IP design vulnerability that is based on spoofed fragments with a correctly guessed IP-ID field. While this attack is similar to our identified fragment duplication attack, the authors focus on the exposure of legitimate IP-IDs in today's network environments and do not propose countermeasures like we do.

Incomplete packets have been found to cause vulnerabilities in commodity operation systems and security appliances [22]. We additionally showed that the transmission of large packets stretched over the reassembly timeout allows to maliciously occupy scarce buffer resources. We counter such attacks with our split buffer approach that is inspired by early queueing strategies for congested ATM switches in case of packet-based communication [35, 9].

With regard to 6LoWPAN, the author in [**?**] claims that implementation deficiencies may enable fragmentation attacks similar to the ones found in IP implementations and that replayed fragmented packets may be a potential security risk. However, in contrast to our work, neither a concrete description of the attacks nor a practical proof is given. Furthermore, the author proposes timestamps or non-cryptographic nonces to mitigate replay attacks. None of these approaches help protecting resource-constrained nodes against our identified attacks as an attacker can simply spoof such information. In [29], the authors analyze the vulnerability of the RPL routing protocol and propose IDS-based countermeasures. Their work is complementary to ours.

Hash chain schemes based on the delayed disclosure of token information such as $\mu TESLA$ [33] have been proposed for the authentication of broadcast messages in resource-constrained environments. While these schemes could also be used to authenticate the sender of fragmented packets, verifying nodes would only be able to authenticate fragments after the delayed token disclosure. Hence, in contrast to our content chaining scheme, verifying nodes would invariably be required to store unauthenticated fragments, even for in-order fragment reception. Moreover, these schemes typically require central coordination or public-key cryptography for hash chain bootstrapping and use long, pre-created hash chains at the sender to compensate the high bootstrapping costs. In contrast, our content chaining scheme does not require special infrastructure or additional cryptography, and keeps the memory overhead at the sender low.

In [28, 15], the authors propose constructions similar to our content chaining scheme for the purpose of secure network programming. However, their approaches require expensive public-key-based operations for the bootstrapping of the hash-chain anchor element, whereas we forgo the RAM, ROM, and CPU overheads of a bootstrapping mechanism. In [12, 25] the above schemes are extended in order to handle packet reordering by employing hash tree-based constructions. These schemes result in considerable token storage requirements at a verifying node for large packets compared to our content chaining scheme.

8. CONCLUSION

In this paper, we analyzed the 6LoWPAN fragmentation mechanism for vulnerabilities at the design level. We focused our analysis on network-internal attackers and an abstract 6LoWPAN network scenario that involves resource-constrained nodes. We revealed two design-level attacks against the 6LoWPAN fragmentation mechanism, the fragment duplication attack and the buffer reservation attack, and showed the susceptibility of 6LoWPAN-enabled nodes with respect to the supported routing mechanisms. The identified attacks are notably cheap allowing an attacker to use tightly resource-constrained nodes for her attacks.

To mitigate these attacks, we propose the content chaining scheme and the split buffer approach with a tailored packet discard strategy. The content chaining scheme allows a node to cryptographically verify that received fragments belong to the same packet on a per-fragment basis. Our split buffer approach fosters direct competition between legitimate senders and an attacker for scarce reassembly buffer resources. In combination with our proposed packet discard strategy, this forces an attacker to invest similar resources for the buffer reservation attack as for a flooding-based attack. Our evaluation confirms the practical existence of the identified attacks and shows that our proposed mechanisms mitigate these at moderate memory and computational costs.

Our work shows that the limited capabilities of resource-constrained nodes and the potentially highly heterogenous resources of IP-enabled devices open new attack vectors for network-internal and network-external attackers. We consider the analysis and the design of secure protocols according to these factors important future work.

Acknowledgments

This work has in parts been funded by the German Federal Ministry for Economic Affairs and Energy under the project funding reference number 01MD11049. The responsibility for the content of this publication lies with the authors.

9. REFERENCES

[1] A. Becher, Z. Benenson, and M. Dornseif. Tampering with motes: real-world physical attacks on wireless sensor networks. In *Proc. of SPC*, 2006.

[2] C. Bormann. Guidance for Light-Weight Implementations of the Internet Protocol Suite. draft-ietf-lwig-guidance-02 (WiP), 2012.

[3] J. W. Bos, O. Özen, and M. Stam. Efficient hashing using the AES instruction set. In *Proc. of CHES*, 2011.

[4] D. Boyle and T. Newe. Security Protocols for Use with Wireless Sensor Networks: A Survey of Security Architectures. In *Proc. of ICWMC*, 2007.

[5] P. G. Bradford and O. V. Gavrylyako. Hash chains with diminishing ranges for sensors. *International Journal of High Performance Computing and Networking*, 2006.

[6] S. Çamtepe and B. Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. *Transactions on Networking*, 2007.

[7] CERT. Advisory CA-1996-26 Denial-of-Service Attack via ping. online @ http://www.cert.org/advisories/CA-1996-26.html, 1996.

[8] CERT. Advisory CA-1997-28 IP Denial-of-Service Attacks. online @ http://www.cert.org/advisories/CA-1997-28.html, 1997.

[9] S. Chan, E. Wong, and K. Ko. Fair packet discarding for controlling ABR traffic in ATM networks . *IEEE Transactions on Communications*, 1997.

[10] I. B. Damgård. A design principle for hash functions. In *Proc. of CRYPTO*, 1989.

[11] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, 1998.

[12] J. Deng, R. Han, and S. Mishra. Secure code distribution in dynamically programmable wireless sensor networks. In *Proc. of IPSN*, 2006.

[13] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes. Powertrace: Network-level power profiling for low-power wireless networks. Technical report, Swedish Institute of Computer Science, 2011.

[14] A. Dunkels, B. Gronvall, and T. Voigt. Contiki – a lightweight and flexible operating system for tiny networked sensors. In *Proc. of IEEE Local Computer Networks*, 2004.

[15] P. K. Dutta, J. W. Hui, D. C. Chu, and D. E. Culler. Securing the deluge Network programming system. In *Proc. of IPSN*, 2006.

[16] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proc. of ACM CCS*, 2002.

[17] R. Gennaro and P. Rohatgi. How to Sign Digital Streams. 1997.

[18] Y. Gilad and A. Herzberg. Fragmentation considered vulnerable: blindly intercepting and discarding fragments. In *Proc. of USENIX Offensive technologies (WOOT)*, 2011.

[19] K. Hartke and O. Bergmann. Datagram Transport Layer Security in Constrained Environments. draft-hartke-core-codtls-02 (WiP), 2012.

[20] C. Hartung, J. Balasalle, R. Han, C. Hartung, J. Balasalle, and R. Han. Node compromise in sensor networks: The need for secure systems. Technical report, University of Colorado at Boulder, 2005.

[21] T. Heer, O. Garcia-Morchon, R. Hummen, S. Keoh, S. Kumar, and K. Wehrle. Security Challenges in the IP-based Internet of Things. *Springer Wireless Personal Communications Journal*, 2011.

[22] K. Hollis. The Rose Attack. online @ http://seclists.org/bugtraq/2004/Mar/351, 2004.

[23] J. Hui and D. Culler. Extending IP to Low-Power, Wireless Personal Area Networks. *Internet Computing, IEEE*, 2008.

[24] R. Hummen, J. H. Ziegeldorf, H. Shafagh, S. Raza, and K. Wehrle. Towards Viable Certificate-based Authentication for the Internet of Things. In *Proc. of ACM HotWiSec*, 2013.

[25] S. Hyun, P. Ning, A. Liu, and W. Du. Seluge: Secure and dos-resistant code dissemination in wireless sensor networks. In *Proc. of IPSN*, 2008.

[26] IEEE. Part 15.4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs). IEEE 802.15.4-2006, 2006.

[27] E. Kim, D. Kaspar, and J. Vasseur. Design and Application Spaces for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs). RFC 6568, 2012.

[28] P. E. Lanigan and P. Narasimhan. Sluice: Secure dissemination of code updates in sensor networks. In *Proc. of ICDCS*, 2006.

[29] A. Le, J. Loo, A. Lasebae, M. Aiash, and Y. Luo. 6LoWPAN: a study on QoS security threats and countermeasures using intrusion detection system approach. *International Journal of Communication Systems*, 2012.

[30] R. C. Merkle. One way hash functions and DES. In *Proc. of CRYPTO*, 1989.

[31] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, 2007.

[32] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou. A survey on jamming attacks and countermeasures in WSNs. *Communications Surveys & Tutorials, IEEE*, 2009.

[33] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. SPINS: security protocols for sensor networks. *Wireless Networks*, 2002.

[34] T. Ptacek and T. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical report, DTIC Document, 1998.

[35] A. Romanow and S. Floyd. Dynamics of TCP traffic over ATM networks. In *Proc. of SIGCOMM*, 1994.

[36] Z. Shelby, K. Hartke, C. Bormann, and B. Frank. Constrained Application Protocol (CoAP). draft-ietf-core-coap-13 (WiP), 2012.

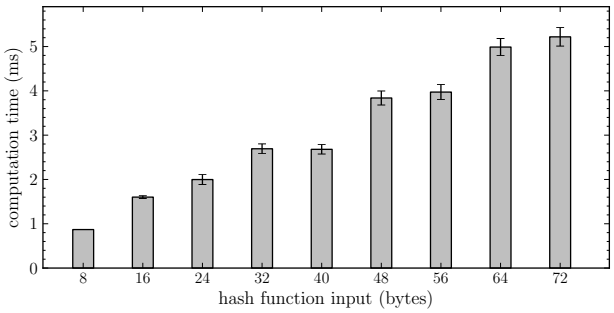


Figure 8: Processing overhead for the generation or the verification of a single content token. The bars represent the mean of the sample for increasing fragment payload sizes. The error bars show the standard error of the mean.

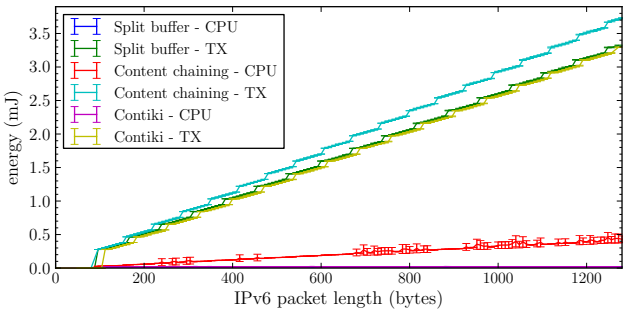


Figure 9: Energy estimation for the computations (CPU) and transmissions (TX) resulting from our proposed defense mechanisms on a sending node. The estimated power consumption for an unmodified Contiki implementation is given as a baseline.

[37] P. Thubert and J. Hui. LoWPAN Fragment Forwarding and Recovery. draft-thubert-6lowpan-simple-fragment-recovery-07 (WiP), 2010.

[38] M. Wilhelm, I. Martinovic, J. B. Schmitt, and V. Lenders. Short paper: reactive jamming in wireless networks: how realistic is the threat? In *Proc. of ACM WiSec*, 2011.

[39] G. Ziemba, D. Reed, and P. Traina. Security Considerations for IP Fragment Filtering. RFC 1858, 1995.

APPENDIX

A. ATTACK NOTIFICATION

To enable a default-off policy for our content chaining scheme, we propose a simple detection and notification mechanism for the fragment duplication attack. This mechanism enables the recipient of a packet with duplicate fragments to request the legitimate sender to apply our content chaining scheme to subsequent fragmented packets. This reactive approach allows for a minimal overhead under normal operation, while protecting the network in case of an attack.

A reassembling node can efficiently detect duplicate fragments by comparing the collected payload in the reassembly buffer to the received fragment. If the payload matches, the received fragment can be silently dropped and the reassembly of the packet can proceed normally. However, if the payload differs, the reassembling node sends an ICMP message with a new, dedicated message type to the packet source notifying about the impending attack. Additionally, the node drops the packet with the duplicate fragments.

When the sender of a fragmented packet receives an ICMP message notifying about a fragment duplication attack, it immediately turns on our content chaining scheme for all further fragmented packets it sends. However, it may turn off the content chaining scheme again after a pre-configured timeout (e.g., after 15 minutes). Hence, during the timespan with active content chaining, an attacker is unable to perform the fragment duplication attack. If the attacker still proceeds with the fragment duplication attack after the timeout expired, the recipient of packets with duplicate fragments triggers another ICMP message and the content chaining scheme is activated again.

Spoofing or dropping of a notification message. As the ICMP message that is used to notify the sender about an impending fragment duplication attack is unauthenticated, an off-path attacker Eve could force the resource-constrained nodes in the 6LoWPAN network to turn on the content chaining scheme by sending a spoofed ICMP message. However, the computation and the packet space overheads she generates this way at the legitimate nodes is equal to the overheads of the content chaining scheme without the notification mechanism. Furthermore, the notification mechanism is designed to turn on the content chaining scheme when an attacker is present in the network. If Eve transmits packets for malicious purposes, this clearly is the case.

Likewise, an on-path attacker Mallory could drop legitimate notification messages in order to prevent the content chaining scheme from being activated. However, if her aim is to block legitimate packets by sending duplicate fragments, dropping of traversing packets is a more effective attack.

B. ENERGY EVALUATION

To get an impression of the energy consumption of our proposed mechanisms, we estimated the power required by the CPU during packet processing and by the radio interface during packet transmission on a sending node. For our estimations, we used the energy estimation utility provided by Contiki [13]. We assume the CPU to require 1.8 mA and the radio interface 19.5 mA per second at a supply voltage of 3 V as indicated in the Tmote Sky data sheet.

As shown by the overlapping CPU energy estimates for the split buffer and an unmodified Contiki implementation (see in Figure 9), the computations involved in the split buffer do not impact the energy consumption. The content chaining scheme, however, increases the energy consumption to a maximum of 0.43 mJ for packets of 1280 byte. These results directly correlate to our run-time performance measurements in Section 6.2.

The steps for the energy estimates of the fragment transmissions in Figure 9 depict the additional transmission overhead resulting from further fragments. The slightly shorter step length of the content chaining scheme results from the fact that less IPv6 packet content can be carried per fragment as 8 byte of fragment payload are taken by the content chaining token. As a result, the sending node has to split the IPv6 packet content into additional 6LoWPAN fragments compared to the 6LoWPAN fragmentation procedure without token information.